Penalty Methods in Discrete Optimization: On the Maximum Number of Threshold Parameters

Dissertation
zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Diplom-Mathematiker Martin Dörnfelder
geboren am 01.10.1984 in Sondershausen

Gutachter

1. Prof. Dr. Ingo Althöfer (Friedrich-Schiller-Universität Jena)
2. Prof. Richard Lorentz, Ph.D. (California State University Northridge, USA)
3. PD Dr. Walter Wenzel (Max Planck Institut für Mathematik in den Naturwissenschaften, Leipzig)

Tag der öffentlichen Verteidigung: 18.12.2009

Contents

Abstract – In German

In der Praxis werden Verfahren der diskreten Optimierung verwendet um für verschiedenste Probleme des Alltags oder der Wirtschaft jeweils die optimale oder eine möglichst gute Lösung zu finden. Dabei ist es oft von Vorteil, mehrere Alternativ-Lösungen zur Auswahl zu haben, um auf eintretende Probleme reagieren zu können. Beispielsweise ist bei einer Reise mit dem Auto die Wegstrecke für die Ermittlung der Reisezeit meistens nicht ausreichend, da Probleme wie Staus und Baustellen die benötigte Zeit für gewisse Streckenabschnitte zum Teil enorm verlängern. Dadurch kann sich die mittels Atlas oder Routenplaner bestimmte Strecke als äußerst ungünstig herausstellen. Hat man sich aber mehrere gute und hinreichend verschiedene Lösungen erzeugt, kann man aus diesen die beste bezüglich der aktuellen Rahmenbedingungen auswählen und so die Reisezeit oft deutlich verkürzen.

In der vorliegenden Arbeit werden zwei Penalty-Methoden zur Erzeugung von Alternativ-Lösungen für diskrete Optimierungs-Probleme vom **Summen-Typ** untersucht.

Das erste und einfachere Verfahren ist die **einfache Penalty Methode**. Hier wird im ersten Schritt die optimale Lösung des Problems bestimmt. In Schritt 2 werden alle Elemente dieser Lösung durch Multiplikation mit einem Faktor $(1+\varepsilon)$ bestraft. Hierbei ist $\varepsilon \geqslant 0$ der sogenannte **Penalty-Parameter**. Im dritten Schritt wird für dieses modifizierte Optimierungs-Problem erneut die optimale Lösung bestimmt – unsere **Penalty-Alternative**.

Das zweite Verfahren ist die **Mutual-Penalty-Methode**. Im Gegensatz zur einfachen Penalty-Methode erfolgt die Berechnung der Alternativen hier nicht nacheinander, sondern simultan. Dabei werden alle Elemente, die in beiden Lösungen vorkommen, erneut mit Hilfe des **Penalty-Parameters** ε bestraft. Elemente, die nur in einer der beiden Lösung auftreten, gehen mit Faktor 1 in die zu minimierende Zielfunktion ein. Die Elemente, die in beiden Lösungen auftreten gehen durch unsere Bestrafung mit Faktor $(2+\varepsilon)$ in die Zielfunktion ein. Ein Paar von Alternativen mit minimalem Gesamtgewicht ist unser Paar von Penalty-Alternativen.

Für diese beiden Verfahren sind schon einige Eigenschaften bekannt. So bilden der Report von Althöfer, Berger und Schwarz [ABS 2002], die Dissertationen von Schwarz [Sch 2003] und Sameith [Sam 2005] sowie die Diplomarbeit [Doe 2008] die Grundlagen der vorliegenden Arbeit. Aus diesen Arbeiten ist bekannt, dass zu allen Alternativen ein konvexes Intervall von Penalty-Parametern angegeben werden kann, für das die Alternative(n) optimal ist. Die Parameter, bei denen ein Optimalitäts-Intervall endet und ein anderes beginnt, heißen **Sprungstellen**.

In dieser Arbeit werden die Penalty-Methoden auf das **Kürzeste-Wege-Problem** und das Problem **minimaler Spannbäume (MST)** angewandt. Dabei werden einige Eigenschaften der erzeugten Alternativen herausgearbeitet. Statt des MST-Problems wird oft die allgemeinere Struktur der **bewerteten Matroide** betrachtet. Die Ergebnisse für bewertete Matroide gelten dann ebenso für das MST Problem. Außerdem werden das **Zuordnungs-Problem** und das **Rundreise-Problem (TSP)** betrachtet um zu zeigen, dass nicht alle Eigenschaften für jedes Summen-Typ-Problem gelten.

Für die einfache Penalty Methode wird untersucht, wie viele Sprungstellen für unsere Optimierungsprobleme maximal existieren können. Zu dieser Fragestellung liefert [Doe 2008] bereits einige Ergebnisse, die hier verallgemeinert werden. Für das Kürzeste-Wege-Problem wird für beliebige Graphen (sogar Multigraphen) mit n Knoten eine obere Schranke von $n^2 - 5n + 10$ für die Anzahl der Sprungstellen bewiesen. Zudem gibt es Beispiele mit $\frac{n^2}{4} - 1$ Sprungstellen. Für bewertete Matroide ergibt sich eine **Struktur-Monotonie**. Dieses Resultat stammt aus [Doe 2008] und zeigt, dass bei jeder Sprungstelle mindestens ein Element, das in der optimalen Lösung enthalten war, gegen ein neues Element ausgetauscht wird. Das entfernte Element tritt in keiner Alternative zu einem größeren Penalty-Parameter mehr auf, das neue Element tritt hingegen in allen dieser Alternativen auf. Mit dieser Struktur-Monotonie wird gezeigt, dass die maximale Anzahl der Sprungstellen eines bewerteten Matroids genau dessen **Rang** entspricht. Damit gibt es beim MST-Problem auf Graphen mit n Knoten maximal $n - 1$ Sprungstellen. Für das Zuordnungs-Problem und das Rundreise-Problem werden Beispiele mit $\frac{n^2}{4} - 1$ bzw. $\frac{(n-1)^2}{4} - 1$ Sprungstellen konstruiert. Die maximale Anzahl der Sprungstellen ist hier also mindestens quadratisch in der Anzahl zuzuordnender Paare bzw. der Anzahl der Knoten des Graphen.

Den Kern dieser Arbeit bildet die Untersuchung der Mutual-Penalty-Methode. Es stellt sich heraus, dass bei bewerteten Matroiden und dem Kürzeste-Wege-Problem nur Elemente bestraft werden können, die auch in der optimalen Lösung enthalten waren. Diese Eigenschaft gilt hingegen nicht beim Zuordnungsproblem und auch nicht beim Rundreiseproblem. Bei den bewerteten Matroiden kann sogar gezeigt werden, dass alle Elemente der optimalen Lösungen auch in allen erzeugten Alternativen auftreten und dass wieder eine Struktur-Monotonie gilt. Damit erhält man wieder den Rang des Matroids als maximale Anzahl der Sprungstellen.
Bei dem Kürzeste-Wege-Problem zeigt sich, dass bei jeder neuen Alternative bestimmte Teilwege erstmals unbestraft auftreten. Mit dieser Eigenschaft ergibt sich eine quadratische obere Schranke von $\frac{n^2}{2} - \frac{n}{2}$. Außerdem existieren wieder Beispiele mit $\frac{n^2}{4} - 1$ Sprungstellen.

Ein großer Nachteil der Mutual-Penalty-Methode war bisher der große Aufwand der zum Erzeugen von Alternativen notwendig war. Wir geben hier für die bewerteten Matroide, das Kürzeste-Wege-Problem und das Zuordnungsproblem Algorithmen an, die asymptotisch genauso schnell sind, wie die entsprechenden Standard-Algorithmen zum Erzeugen einer einzelnen Lösung.

Acknowledgements

The writing of this dissertation has been one of the most significant academic challenges I have ever had to face. Without the support, patience and guidance of the following people, this research would not have been finished so fast and good. It is to them that I owe my deepest gratitude.

- Prof. Dr. Ingo Althöfer who undertook to act as my supervisor. His wisdom, knowledge and commitment to the highest standards inspired and motivated me.
- My friends and colleagues, Rico Walter and Jakob Erdmann, who inspired my research by many helpful discussions and who acted as proofreaders of this thesis.
- Markus Fahsel and Justin Neville who acted as proofreaders and gave me many helpful hints to improve the quality of this thesis.
- Peter and Ingrid Dörnfelder, my parents, who have always supported, encouraged and believed in me.

Chapter 1

Introduction

In discrete optimization one is typically interested in finding the optimal or one good solution for a given optimization problem. But in practice it is sometimes better to generate not only one good solution but two or more good alternatives.

A typical approach in mathematics is to translate a given real world problem into a model problem in order to solve it with known techniques. Then, the so found solution will be translated back to a solution of the real world problem. Since models typically do not take all facts into account the final result is often not going to be the optimal solution of the real world problem. An example: If we are trying to generate the fastest route for a trip by car, we are faced with exactly this kind of problem. By using a map or vehicle routing program, the solution that is found will only consider the length and kinds of roads, but not directly the real time needed to pass each of them. Normally this approach works quite well, but exceptional circumstances such as construction zones or traffic jams are not taken into consideration. So we finally do not get the optimal solution. One way out of this dilemma is a generalization of the used model. But this is often impractical. A better approach is the generation of several alternatives that all differ from each other and all give a decent route. Knowing the real traveling times needed for each of our route's section, it is now possible to choose the best final solution amongst the generated alternatives.

In this thesis two penalty methods for the generation of alternative solutions for **sum type problems** are investigated.

The first approach is the **simple penalty method**. In the first step, it generates the optimal solution for the given optimization problem. Then all parts of this solution will be multiplied by the factor $(1 + \varepsilon)$ and all other edge weights remain unmodified. The parameter $\varepsilon \geqslant 0$ is called **penalty parameter**. We get our **penalty alternative** by generating the optimal solution for the punished problem.

The second approach is the **mutual penalty method**. Again, all elements which are part of both alternatives are punished with extra weights. In difference to the simple penalty method, we do not generate the best solution and a good alternative for it, but we generate both alternatives simultaneously. The weights of elements which are part of exactly one of the alternatives are summed up in our objective function. But the weights of elements which are used by both alternatives are not summed up twice, but $(2+\varepsilon)$-times. A pair of alternatives with a minimum punished weight sum is our pair of alternatives.

For both methods some results already exist. In this way, the report of Althöfer, Berger and Schwarz [ABS 2002], the doctoral dissertations of Schwarz [Sch 2003] and Sameith [Sam 2005] as well as the diploma thesis [Doe 2008] give some basics for this thesis. It is known that there exists a convex interval of penalty parameters for every alternative so that the alternatives are optimal for all parameters of this interval and for no other parameter. The parameters that are endpoints of one optimality interval and starting points of another optimality interval are called **threshold parameters**.

In this thesis we formulate some properties of the generated alternatives. Specifically, the penalty methods will be applied to **shortest path problems** and **minimum spanning tree problems (MST)**. Sometimes the structure of **valuated matroids** will be used to give properties of the MST problem. For differentiation we also give some results for the **assignment problem** and the **traveling salesperson problem (TSP)**.

In Chapter 2 we introduce some basic notations and definitions. First, some notations of directed and undirected graphs and the big-O-notation will be recapitulated. Then, it will be explained what **sum type problems** are. After that, we recapitulate some methods to generate alternative solutions for sum type problems. Here we also formulate some general properties of the penalty alternatives. As we want to give the results for the MST problem in a more general context, the structure of valuated matroids will be defined. Finally, we prepare the formulation of algorithms for the generation of mutual penalty alternatives by defining the minimum cost flow problem.

In Chapter 3 we investigate the simple penalty method. Regarding the shortest path problem on directed acyclic graphs it was shown in [Doe 2008] that at most $\frac{n^2}{2} - \frac{3n}{2} + 2$ threshold parameters are possible. Here and in all other contexts, n is the number of vertices in the underlying graph. In this chapter we leave directed acyclic graphs and investigate arbitrary multigraphs. For this general case we give an upper bound of $n^2 - 5n + 10$ and a lower bound of $\frac{n^2}{4} - 1$ for the maximum number of threshold parameters. For the MST problem (and more generally for the valuated

matroids) there exists a structural monotonicity. This means that an element e of the optimal solution is replaced by a new element f at every threshold parameter. For all larger penalty parameters the element e will not be part of the penalty alternative and f will be part of all of these alternatives. With this monotonicity we prove that the maximum number of threshold parameters is exactly $n-1$ (or the **rank** of the valuated matroid).

The main part of this thesis is Chapter 4. Here we take a look at the mutual penalty method. One experimental result of Sameith [Sam 2005, p. 40] is that the mutual penalty method generates alternatives which are as good as those for the simple penalty method. Unfortunately, it was very time intensive to generate those mutual penalty alternatives. As we generate two alternatives simultaneously and not one after the other, it is difficult to see what this method really does. To reduce these difficulties we formulate fast algorithms for the generation of mutual penalty alternatives and reveal some nice properties for the shortest path problem and the valuated matroids. The most surprising property of mutual penalty alternatives for the shortest path problem and valuated matroids is that all common elements of both alternatives are also part of the optimal solution. This property does not hold for the assignment problem and the TSP.

Regarding the valuated matroids we show that all elements of the optimal solution are part of our pair of mutual penalty alternatives. Again, we get a structural monotonicity. This structural monotonicity gives the result that the maximum number of threshold parameters is exactly the rank of the matroid. Regarding the shortest path problem we show that both alternatives use their common vertices in the same order and that vertices that lie on the shortest $s-t$-path are used in the same order in all pairs of alternatives. With the help of these properties we give an upper bound of $\frac{n^2}{2}-\frac{n}{2}$ for the number of threshold parameters. As we also give an example with $\frac{n^2}{4}-1$ threshold parameters, the maximum number of threshold parameters lies in the class $\Theta(n^2)$.

In Chapter 5 we conclude the results of this thesis. Here we also compare the bounds for the number of threshold parameters for the simple penalty method and the mutual penalty method with bounds for the **k-best method** and the **generalized penalty method**.

In Chapter 6 we discuss some open questions for future research.

As some definitions, remarks and theorems of Chapter 2 will be used quite often in the Chapters 3 and 4, it is recommended to read Chapter 2 first. After that, Chapters 3 and 4 can be read independently from each other.

Chapter 2

Basics

2.1 Basic Notations and Definitions

Before starting to introduce methods for the generation of alternative solutions, we give some notations.

In this thesis, mainly graph problems will be investigated. In this context the symbols $G = (V, E)$ will be used as well for directed as for undirected graphs with vertex set V and edge set E. These sets have cardinalities $|V| = n$ and $|E| = m$.

In some cases it does not make sense to give the bounds or runtimes of algorithms as exact values. Thus, we use $f \in EXP(n)$ if f is exponentially increasing in n and in other cases we use the big-O-notation which is formulated in the following definition.

Definition 2.1.1 (Big-O-Notation).
Let f and g be two integer-valued functions on $\mathbb{N}$. Then we define:

1. $f(n) \in \boldsymbol{O(g(n))}$ *means that there are positive constants c and n_0, such that $0 \leqslant f(n) \leqslant c \cdot g(n)$ for all $n \geqslant n_0$. The values of c and n_0 must be fixed for the function f and must not depend on n.*

2. $f(n) \in \boldsymbol{\Omega(g(n))}$ *means that there are positive constants c and n_0, such that $0 \leqslant c \cdot g(n) \leqslant f(n)$ for all $n \geqslant n_0$. The values of c and n_0 must be fixed for the function f and must not depend on n.*

3. $f(n) \in \boldsymbol{\Theta(g(n))}$ *means that there are positive constants c_1, c_2, and n_0, such that $0 \leqslant c_1 \cdot g(n) \leqslant f(n) \leqslant c_2 \cdot g(n)$ for all $n \geqslant n_0$. The values of c_1, c_2, and n_0 must be fixed for the function f and must not depend on n.*

Throughout this thesis we investigate optimization problems for which it is easy to determine the weight of a solution. A big class of these problems are sum type problems. There, the weight of a solution is the sum of the weights of all building blocks. Summing up some numbers is quite easy and so we can determine very fast how good the different alternative solutions are and which one is the best amongst them.

Definition 2.1.2 (Sum Type Problem).
Let E be an arbitrary finite set and S a subset of the power set $S \subseteq \mathcal{P}(E)$ of E. Then E is called the ***base set*** *and all elements of S are* ***feasible solutions****. Let $w : E \to \mathbb{R}$ be a* ***weight function*** *on E. For all $B \in S$ we set $w(B) = \sum_{e \in B} w(e)$. Then we call the problem $\min_{B \in S} w(B)$ a* ***Sum Type Problem****.*

Remark 2.1.3.
We generalize the definition of sum type problems with the help of the following two properties:

- In some situations we also use the weight of subsets of E which are not solutions of the given optimization problem. Thus, we also define
$$w(B) = \sum_{e \in B} w(e) \text{ for all } B \in \mathcal{P}(E).$$
With this generalization of sum type problems we can also calculate the weights of subpaths and other subsets of solutions.

- We can also formulate every maximization problem as a minimization problem by defining $w' := -w$.

  Thus, we also denote special maximization problems as sum type optimization problems. This property will be especially used for the structure of valuated matroids.

There exist many famous problems which fulfill the sum type property. As a simple sum type problem we investigate the minimum spanning tree problem (MST). One advantage of this problem is that all results which are given by the structure of valuated matroids also hold for minimum spanning trees. Another advantage is that there exist very fast algorithms for the generation of minimum spanning trees. The shortest path problem does not fulfill the matroid axioms, but it might be the most intuitive sum type problem. The assignment problem is algorithmically harder to solve and the TSP is the hardest of the problems under investigation. In this thesis we will mainly investigate the MST problem and the shortest path problem.

Minimum Spanning Tree Problem (MST)

Consider an undirected graph $G = (V, E)$ and a positive weight function $w : E \to \mathbb{R}^+$ on the set of edges E of G. A spanning tree of that graph is a subgraph which is a tree (it does not contain circuits) and connects all the vertices. A minimum spanning tree (MST) in G is a spanning tree with minimum weight amongst all spanning trees in G. By choosing E as base set and S as the family of all edge sets which represent spanning trees in G we get an optimization problem of sum type.

In Section 2.3 the valuated matroids will be introduced. They are a generalization of the minimum spanning tree problem and some other optimization problems. Often, we use valuated matroids to prove properties of alternatives. These results hold for all problems which fulfill the matroid properties.

Shortest Path Problem

Consider a directed graph $G = (V, E)$ and a positive weight function $w : E \to \mathbb{R}^+$ on the edge set E of G. Let s and t be two distinguished vertices of G. Which is the shortest path from the start vertex s to the target vertex t in G and what is its length? By choosing E as base set and S as the family of all edge sets which represent paths from s to t in G we get a sum type problem.

Assignment Problem

Consider a number of agents $A = \{A_1, A_2, \ldots, A_n\}$ and a number of tasks $T = \{T_1, T_2, \ldots, T_n\}$. Agent A_i can be assigned to perform any task T_j, incurring some cost $c(i, j)$. It is required to perform all tasks by assigning exactly one agent to it in a way that the total cost of the assignment is minimized. By choosing E as the cross product $A \times T$ and S as the family of all subsets of E which represent a feasible assignment of agents to tasks we get an optimization problem of sum type.

Traveling Salesperson Problem (TSP)

Consider a set of n cities $C = \{C_1, C_2, \ldots, C_n\}$ and a distance function $d : C \times C \to \mathbb{R}^+$. Which is the shortest closed tour through all n cities? By choosing E as the cross product $C \times C$ and S as the family of edge sets which represent a closed tour through all n cities we get an optimization problem of sum type.

2.2 Methods for the Generation of Alternative Solutions

As motivated in the introduction, often it is very helpful to have not only one good solution, but rather to have two or more "good" alternatives at hand. But what do we understand under "good" alternatives? We cannot define what they are, but we give the following two criteria which "good" alternative solutions should fulfill:

(i) An alternative solution should be good in respect to the objective function, otherwise it would not make any sense to choose it.

(ii) The alternative solutions should not be too similar to each other, otherwise they would not be true alternatives. In the case of micro mutations many disadvantages of one solution are quite likely also in the other one.

There exist many different methods to generate alternative solutions. First of all, we recapitulate the k-best method which was developed by Bellman and Kalaba [BK 1958]. Then we introduce four penalty methods which are given by [ABS 2002] and [Sch 2003]. Here Althöfer, Berger and Schwarz [ABS 2002] showed that we can control the way the conflicting criteria (i) and (ii) will be fulfilled by the choice of the penalty parameter.

The k-best method and the generalized penalty method will be used in Chapter 5 to compare the known results for these approaches with the new results for the simple penalty method and the mutual penalty method.

2.2.1 k-best Method

Already in the 1950's the problem of generating alternative solutions was studied. Bellman and Kalaba [BK 1958] formulated the k-best Method which is the most intuitive approach to generate alternative solutions. The k-best method for some $k \in \mathbb{N},\ k > 1$ does not generate only the best solution, but also the k-th best solution as an alternative one. So we get the second best solution for $k = 2$, the third best solution for $k = 3$ and so on. Using this approach we get really good results concerning our criterion (i). But practical applications show that this method does not produce practicable results concerning our criterion (ii). Typically, the second, third, fourth, ... best alternatives are just **micro mutations** of the optimal solution. This was shown by Schwarz [Sch 2003, p. 6] experimentally. Because of that, we need other methods for the generation of alternatives. In the following two sections we introduce the **simple penalty method**, the **mutual penalty method** and generalizations of these approaches.

2.2.2 Simple Penalty Method

Instead of generating the k-best alternative for any k, we punish all elements of the optimal solution with extra weights. With these punishments we make many of these elements unattractive for our alternatives. It turns out that we can improve the difference between our two generated solutions by increasing the punishment and that we can improve the weight of the generated alternative by decreasing the punishment. In this way we can control how good our criteria (i) and (ii) are fulfilled.

Now we define the method described above formally.

Definition 2.2.1 (Simple Penalty Method).
*Let E be an arbitrary finite set and $S \subseteq \mathcal{P}(E)$ be the set of feasible solutions in E. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E and B_0 be an optimal solution for the problem $\min\limits_{B \in S} w(B)$. For every $\varepsilon > 0$ we get a **penalty alternative** B_ε as an optimal solution of the modified problem*

$$B_\varepsilon = \min_{B \in S} \left[w(B) + \varepsilon \cdot w(B \cap B_0) \right].$$

Additionally, we define the solution B_∞ of the problem as

$$B_\infty = \operatorname*{lexmin}_{B \in S} \left(w(B \cap B_0), w(B) \right).$$

This means that B_∞ has a minimum weight intersection with B_0 and among all of these solutions B_∞ is one with minimum weight $w(B)$.

The simple penalty alternatives can be generated very easily. In the first step, we generate the optimal solution for the basic optimization problem. This can be done with standard algorithms. Then, we multiply the weights of all elements that are part of this solution by the factor $(1 + \varepsilon)$. Finally, we generate the optimal solution for the modified problem and get our penalty alternatives. Also in this last step, we can use standard algorithms.

As many basic properties hold in a more general context, we generalize the simple penalty method and name this new approach **generalized penalty method**.

Definition 2.2.2 (Generalized Penalty Method).
*Let E be an arbitrary finite set and $S \subseteq \mathcal{P}(E)$ be the set of feasible solutions in E. Let $w : E \to \mathbb{R}^+$ be a positive weight function and $p : E \to \mathbb{R}^+$ be a positive **penalty function** on E. For all $\varepsilon > 0$ we get a **penalty alternative** B_ε as an optimal solution of the problem $B_\varepsilon = \min\limits_{B \in S} \left[w(B) + \varepsilon \cdot p(B) \right]$.*

Additionally we define the solution B_∞ of the problem as:

$$B_\infty = \operatorname*{lexmin}_{B \in S} \left(p(B), w(B) \right).$$

Here "lexmin" means lexicographic minimization. B_∞ has a minimal penalty value $p(B)$ and among all these solutions B_∞ is one with minimal weight $w(B)$.

Like for the simple penalty method we can use standard algorithms for the generation of the penalty alternatives.

The generalized penalty method (and especially the simple penalty method) has some properties which are important for the investigation of the simple penalty method. These properties are given by the following two theorems.

Theorem 2.2.3 (Althöfer, Berger, Schwarz [ABS 2002]).
Let $w, p : E \to \mathbb{R}^+$ be positive weight functions on E. Let B_ε be defined for all $\varepsilon > 0$ according to Definition 2.2.2. Then the following statements hold:

(i) $p(B_\varepsilon)$ *is weakly monotonically decreasing in* ε.

(ii) $w(B_\varepsilon)$ *is weakly monotonically increasing in* ε.

Proof
Let δ and ε be two arbitrary nonnegative real numbers with $0 \leqslant \delta < \varepsilon$. As B_δ and B_ε are optimal alternatives for the penalty parameters ε and δ, the following inequalities hold:

(i) In the case $\varepsilon < \infty$ we have

$$w(B_\varepsilon) + \varepsilon \cdot p(B_\varepsilon) \leqslant w(B_\delta) + \varepsilon \cdot p(B_\delta), \tag{2.2.1}$$
$$w(B_\delta) + \delta \cdot p(B_\delta) \leqslant w(B_\varepsilon) + \delta \cdot p(B_\varepsilon). \tag{2.2.2}$$

Subtracting (2.2.2) from (2.2.1) we get

$$\begin{aligned} & (\varepsilon - \delta) \cdot p(B_\varepsilon) \leqslant (\varepsilon - \delta) \cdot p(B_\delta) \\ \Leftrightarrow \quad & p(B_\varepsilon) \leqslant p(B_\delta). \end{aligned} \tag{2.2.3}$$

In the case $\varepsilon = \infty$, inequality (2.2.3) follows directly from the definition of B_∞.

(ii) Subtracting (2.2.3) multiplied by δ from (2.2.2) we get $w(B_\varepsilon) \geqslant w(B_\delta)$. ■

Theorem 2.2.4 (Schwarz [Sch 2003, pp. 16-17]).
If B is an ε-optimal solution then there exists an interval $OPT_B = [\varepsilon_i, \varepsilon_j]$ with $\varepsilon_i, \varepsilon_j \in \mathbb{R} \cup \{\infty\}$, so that B is optimal for every penalty parameter $\varepsilon \in OPT_B$ and for no other parameters.

Proof
Assume B is optimal for the penalty parameters $\varepsilon_i, \varepsilon_j \in \mathbb{R} \cup \{\infty\}$ with $\varepsilon_i < \varepsilon_j$.

In the case $\varepsilon_j < \infty$ we have

$$\begin{aligned} w(B) + \varepsilon_i \cdot p(B) &\leqslant w(B') + \varepsilon_i \cdot p(B') \text{ for all } B' \in S, && (2.2.4)\\ w(B) + \varepsilon_j \cdot p(B) &\leqslant w(B') + \varepsilon_j \cdot p(B') \text{ for all } B' \in S. && (2.2.5) \end{aligned}$$

For an intermediate value $\varepsilon \in (\varepsilon_i, \varepsilon_j)$ we multiply (2.2.4) by $\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} > 0$ and (2.2.5) by $\frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} > 0$. Adding up these two inequalities yields

$$\begin{aligned} & \left(\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} + \frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i}\right) \cdot w(B) + \left(\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_i + \frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_j\right) \cdot p(B) \\ \leqslant \; & \left(\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} + \frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i}\right) \cdot w(B') + \left(\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_i + \frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_j\right) \cdot p(B') \\ & \qquad \text{for all } B' \in S \\ \Leftrightarrow \quad & w(B) + \varepsilon \cdot p(B) \leqslant w(B') + \varepsilon \cdot p(B') \text{ for all } B' \in S. \end{aligned}$$

This shows that B is also optimal for all intermediate penalty parameters $\varepsilon \in (\varepsilon_i, \varepsilon_j)$.

In the case $\varepsilon_j = \infty$ we have

$$\begin{aligned} w(B) + \varepsilon_i \cdot p(B) &\leqslant w(B') + \varepsilon_i \cdot p(B') \text{ for all } B' \in S, && (2.2.6)\\ p(B) &\leqslant p(B') \text{ for all } B' \in S. && (2.2.7) \end{aligned}$$

For a value $\varepsilon > \varepsilon_i$ we multiply (2.2.7) by $\varepsilon - \varepsilon_i > 0$ and add (2.2.6). Thus we have

$$w(B) + \varepsilon \cdot p(B) \leqslant w(B') + \varepsilon \cdot p(B') \text{ for all } B' \in S. \qquad (2.2.8)$$

This shows that B is also optimal for all penalty parameters $\varepsilon > \varepsilon_i$.

Thus, the set of penalty parameters for which an alternative is optimal is convex. This shows that we can find an interval $OPT_B = [\varepsilon_i, \varepsilon_j]$ such that B is optimal for all parameters $\varepsilon \in OPT_B$ and for no other parameters. ■

Definition 2.2.5 (Threshold Parameter).
*The penalty parameters $\varepsilon_i, \varepsilon_j$ from Theorem 2.2.4 which are positive and finite, are called **threshold parameters**. For an increasing penalty parameter ε the generated solution is switching to an other one at these threshold parameters.*

2.2.3 Mutual Penalty Method

The mutual penalty method is another approach to generate alternative solutions by using penalties. This method was introduced in [Sch 2003, pp. 20-27] and was named "Linear Programming Penalty Method". Sameith [Sam 2005, p. 15] renamed it to "Mutual Penalty Method". As there exist other algorithms which do not use linear programs to generate the alternatives, we also call it "Mutual Penalty Method". As done by the simple penalty method, this approach also punishes all elements which are part of both solutions. The difference is that we do not generate an optimal solution and then a good alternative for it. Here, we generate two alternatives simultaneously according to the following definition:

Definition 2.2.6 (Mutual Penalty Method).
*The **mutual penalty method** generates a pair of **mutual penalty alternatives** $B_{1(\varepsilon)}$ and $B_{2(\varepsilon)}$ simultaneously. These solutions are defined as*

$$\left\{B_{1(\varepsilon)}, B_{2(\varepsilon)}\right\} = \min_{B_1, B_2 \in S} \left[w\left(B_1\right) + w\left(B_2\right) + \varepsilon \cdot w\left(B_1 \cap B_2\right)\right].$$

Additionally, we define the solutions $\left\{B_{1(\infty)},\ B_{2(\infty)}\right\}$ of the problem as:

$$\left\{B_{1(\infty)},\ B_{2(\infty)}\right\} = \underset{\{B_1, B_2\} \in S}{lexmin} \left(w\left(B_1 \cap B_2\right), w\left(B_1\right) + w\left(B_2\right)\right).$$

This means that the solutions $\left\{B_{1(\infty)},\ B_{2(\infty)}\right\}$ have a minimal weight intersection with each other and among all these solutions $\left\{B_{1(\infty)},\ B_{2(\infty)}\right\}$ is one with a minimal sum of weights $w\left(B_1\right) + w\left(B_2\right)$.

One big advantage of this method is that it can be easily generalized to generate any number of alternatives. Here we define the k-mutual penalty method which generates k alternatives simultaneously. In this thesis the case $k = 2$ will be investigated, but the basics are not more difficult for the general case. In Chapter 6 we formulate open problems concerning the k-mutual penalty method.

Definition 2.2.7 (k-Mutual Penalty Method).
*The k-**mutual penalty method** generates a k-set of **mutual penalty alternatives** $B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}$ simultaneously. As for the generalized penalty method we use a nonnegative penalty function p. Here p depends on the generated alternatives in any way. The **k-mutual penalty alternatives** are defined as follows:*

$$\left\{B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right\} = \min_{B_1, B_2, \ldots, B_k \in S} \left\{\left[\sum_{i=1}^{k} w\left(B_i\right)\right] + \varepsilon \cdot p\left(B_1, B_2, \ldots, B_k\right)\right\}.$$

The alternatives $\left\{B_{1(\infty)},\ B_{2(\infty)}, \ldots, B_{k(\infty)}\right\}$ are defined analogously to the previous definitions:

$$\left\{B_{1(\infty)},\ B_{2(\infty)}, \ldots,\ B_{k(\infty)}\right\} = \underset{\{B_1, B_2, \ldots, B_k\} \in S}{lexmin} \left(p\left(B_1, B_2, \ldots, B_k\right), \sum_{i=1}^{k} w\left(B_i\right)\right).$$

The Theorems 2.2.3 and 2.2.4 hold analogously for the k-mutual penalty method:

Theorem 2.2.8.
Let $w : E \to \mathbb{R}^+$ be a positive weight function on E. Let $\left\{B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right\}$ be defined for all $\varepsilon > 0$ according to Definition 2.2.7. Then the following statements hold:

(i) $p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right)$ *is weakly monotonically decreasing in* ε.

(ii) $\sum_{i=1}^{k} w\left(B_{i(\varepsilon)}\right)$ *is weakly monotonically increasing in* ε.

Proof
Let δ and ε be two arbitrary nonnegative real numbers with $0 \leqslant \delta < \varepsilon$. For $\left\{B_{1(\delta)}, B_{2(\delta)}, \ldots, B_{k(\delta)}\right\}$ and $\left\{B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right\}$ the following inequalities hold:

(i) In the case $\varepsilon < \infty$ we have

$$\sum_{i=1}^{k} w\left(B_{i(\varepsilon)}\right) + \varepsilon \cdot p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right) \leqslant \sum_{i=1}^{k} w\left(B_{i(\delta)}\right) + \varepsilon \cdot p\left(B_{1(\delta)}, B_{2(\delta)}, \ldots, B_{k(\delta)}\right), \tag{2.2.9}$$

$$\sum_{i=1}^{k} w\left(B_{i(\delta)}\right) + \delta \cdot p\left(B_{1(\delta)}, B_{2(\delta)}, \ldots, B_{k(\delta)}\right) \leqslant \sum_{i=1}^{k} w\left(B_{i(\varepsilon)}\right) + \delta \cdot p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right). \tag{2.2.10}$$

Subtracting (2.2.10) from (2.2.9) we get

$$\begin{aligned} & (\varepsilon - \delta) \cdot p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right) \leqslant (\varepsilon - \delta) \cdot p\left(B_{1(\delta)}, B_{2(\delta)}, \ldots, B_{k(\delta)}\right) \\ \Leftrightarrow \quad & p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}, \ldots, B_{k(\varepsilon)}\right)) \leqslant p\left(B_{1(\delta)}, B_{2(\delta)}, \ldots, B_{k(\delta)}\right). \end{aligned} \tag{2.2.11}$$

In the case $\varepsilon = \infty$ Inequality (2.2.11) follows directly from the definition of $\left\{B_{1(\infty)}, B_{2(\infty)}, \ldots, B_{k(\infty)}\right\}$.

(ii) Subtracting (2.2.11) multiplied by δ from (2.2.10) we get

$$\sum_{i=1}^{k} w\left(B_{i(\varepsilon)}\right) \geqslant \sum_{i=1}^{k} w\left(B_{i(\delta)}\right).$$

■

Theorem 2.2.9.
If $\{B_1, B_2, \ldots, B_k\}$ is an ε-optimal solution set then there exists an interval $OPT_{\{B_1, B_2, \ldots B_k\}} = [\varepsilon_i, \varepsilon_j]$ with $\varepsilon_i, \varepsilon_j \in \mathbb{R} \cup \{\infty\}$, so that $\{B_1, B_2, \ldots, B_k\}$ is optimal for every penalty parameter $\varepsilon \in OPT_{\{B_1, B_2, \ldots B_k\}}$ and for no other parameters.

Proof
Assume $\{B_1, B_2, \dots B_k\}$ is optimal for the penalty parameters $\varepsilon_i, \varepsilon_j \in \mathbb{R} \cup \infty$ with $\varepsilon_i < \varepsilon_j$.
In case of $\varepsilon_j < \infty$ we have

$$\left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \varepsilon_i \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \varepsilon_i \cdot p\left(\{B'_1, B'_2, \dots B'_k\}\right)$$
$$\text{for all } B'_1, B'_2, \dots B'_k \in S, \tag{2.2.12}$$

$$\left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \varepsilon_j \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \varepsilon_j \cdot p\left(\{B'_1, B'_2, \dots B'_k\}\right)$$
$$\text{for all } B'_1, B'_2, \dots B'_k \in S. \tag{2.2.13}$$

For an intermediate value $\varepsilon \in (\varepsilon_i, \varepsilon_j)$ we multiply (2.2.12) by $\frac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} > 0$ and (2.2.13) by $\frac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} > 0$. Adding up these two inequalities gives

$$\begin{aligned}
&\left(\tfrac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} + \tfrac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i}\right) \cdot \left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \left(\tfrac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_i + \tfrac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_j\right) \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \\
\leqslant\ &\left(\tfrac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} + \tfrac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i}\right) \cdot \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \left(\tfrac{\varepsilon_j - \varepsilon}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_i + \tfrac{\varepsilon - \varepsilon_i}{\varepsilon_j - \varepsilon_i} \cdot \varepsilon_j\right) \cdot p\left(\{B'_1, B'_2, \dots, B'_k\}\right) \\
&\qquad \text{for all } B'_1, B'_2, \dots, B'_k \in S \\
\Leftrightarrow\ &\left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \varepsilon \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \varepsilon \cdot p\left(\{B'_1, B'_2, \dots B'_k\}\right) \\
&\qquad \text{for all } B'_1, B'_2, \dots, B'_k \in S.
\end{aligned}$$

This shows that $\{B_1, B_2, \dots B_k\}$ is also optimal for all intermediate penalty parameters $\varepsilon \in (\varepsilon_i, \varepsilon_j)$.

In case of $\varepsilon_j = \infty$ we have

$$\left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \varepsilon_i \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \varepsilon_i \cdot p\left(\{B'_1, B'_2, \dots B'_k\}\right)$$
$$\text{for all } B'_1, B'_2, \dots B'_k \in S, \tag{2.2.14}$$

$$p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant p\left(\{B'_1, B'_2, \dots B'_k\}\right)$$
$$\text{for all } B'_1, B'_2, \dots B'_k \in S. \tag{2.2.15}$$

For a value $\varepsilon > \varepsilon_i$ we multiply (2.2.15) by $\varepsilon - \varepsilon_i > 0$ and add (2.2.14).

$$\left(\sum_{l=1}^{k} w\left(B_{l(\varepsilon)}\right)\right) + \varepsilon \cdot p\left(\{B_1, B_2, \dots B_k\}\right) \leqslant \left(\sum_{l=1}^{k} w\left(B'_{l(\varepsilon)}\right)\right) + \varepsilon \cdot p\left(\{B'_1, B'_2, \dots B'_k\}\right)$$
$$\text{for all } B'_1, B'_2, \dots B'_k \in S. \qquad (2.2.16)$$

This shows that $\{B_1, B_2, \dots B_k\}$ is also optimal for all penalty parameters $\varepsilon > \varepsilon_i$.

Thus, the set of penalty parameters for which an alternative is optimal, is convex. Because of this convexity we can find an interval $OPT_{\{B_1,B_2,\dots B_k\}} = [\varepsilon_i, \varepsilon_j]$ such that $\{B_1, B_2, \dots B_k\}$ is optimal for every penalty parameter $\varepsilon \in OPT_{\{B_1,B_2,\dots B_k\}}$ and for no other parameters. ■

Remark 2.2.10.
By setting $k = 2$,

$$p(e) := \begin{cases} w(e), & \text{if } e \in B_{1(\varepsilon)} \cap B_{2(\varepsilon)}, \\ 0, & \text{otherwise} \end{cases}$$

and

$$p\left(B_{1(\varepsilon)}, B_{2(\varepsilon)}\right) := \sum_{e \in B_{1(\varepsilon)} \cap B_{2(\varepsilon)}} p(e)$$

we get the mutual penalty method. Thus, the Theorems 2.2.8 and 2.2.9 analogously hold for the mutual penalty method.

Remark 2.2.11.
In this thesis we denote the generation of mutual penalty alternatives for the shortest path problem as mutual shortest path problem. Analogously, we denote mutual matroids, mutual MST, mutual assignments and mutual TSP.

2.2.4 Some More Notations for the Penalty Methods

The following definitions and remarks will be used for the simple penalty method and for the mutual penalty method. For sake of simplicity we motivate them with the help of the simple penalty method.

Definition 2.2.12 (New Element, Old Element).
*An element $x \in B$ is called **new** at a penalty parameter ε, if B is optimal for the penalty parameter ε and x is not part of any alternative B' which is optimal for any penalty parameter ε' with $0 \leqslant \varepsilon' < \varepsilon$. Otherwise the element x is called **old**.*

Remark 2.2.13.
In many cases we use the concrete terms "new edge", "new gap", ... instead of "new element".

Definition 2.2.14 (New Alternative, Old Alternative).
*An alternative B is called **new** at a penalty parameter ε, if B is optimal for the penalty parameter ε and not optimal for any penalty parameter ε' with $0 \leqslant \varepsilon' < \varepsilon$. Otherwise the alternative B is called **old**.*

Definition 2.2.15 (Newest Element).
*An element $e \in B$ is called a **newest** element of B if $e \in B$, e is new at the penalty parameter ε and all other elements $f \in B$ are new at any penalty parameter ε' with $0 \leqslant \varepsilon' \leqslant \varepsilon$.*

Definition 2.2.16 (Newest Alternative).
*An alternative B is called a **newest** alternative of a set of alternatives $\mathcal{P}$, if it was new for the largest penalty parameter amongst all alternatives in the set $\mathcal{P}$.*

New(est) elements and alternatives do not have to be unique. An alternative can contain more than one new(est) element and at a threshold parameter ε^* there can be more than one alternative new (if both have the same punished and the same unpunished weight).

Remark 2.2.17.
If B is an optimal alternative for an arbitrary penalty parameter ε and $e \in B$ is new at ε, then the alternative B is new at the penalty parameter ε.

On the one hand B is optimal for ε as proposed. On the other hand B cannot be optimal for any penalty parameter ε' with $\varepsilon' < \varepsilon$, because the element e is new at ε. As e is new at ε, there does not exist any alternative which is optimal for any penalty parameter ε' with $\varepsilon' < \varepsilon$ and contains e. As B contains e, B cannot be optimal for any penalty parameter ε' with $\varepsilon' < \varepsilon$.

Remark 2.2.18.
A new alternative has a smaller punished weight than all alternatives that were optimal for smaller penalty parameters. The monotonicity theorem (Theorem 2.2.3) shows that the punished weight of the new alternative cannot be larger than the punished weight of any alternative which was optimal for a smaller penalty parameter. If the punished weights of two alternatives B_1 and B_2 are equal, then the punished weights of both alternatives are linear functions which are either equal (if $w(B_1) = w(B_2)$) or parallel. Thus, B_1 and B_2 are optimal at the same interval of penalty parameters or one of them is better than the other one for all penalty parameters. In both cases there does not exist any threshold parameter at which the generated alternative switches from B_1 to B_2.

2.2.5 Finding all Threshold Parameters

Schwarz [Sch 2003, pp. 15-18] gave an algorithm to find a sequence $\mathcal{B}$ of ε-optimal solutions covering all $\varepsilon \geqslant 0$ for a given sum type problem. We modify it a little to get the set of threshold parameters as output. This algorithm works as well for the simple penalty method as for the mutual penalty method. With both approaches we generate two alternatives (B_0 and B_ε for the simple penalty method and $B_{\varepsilon(1)}$ and $B_{\varepsilon(2)}$ for the mutual penalty method). For sum type problems we denote $w(B)$ as the sum of the unpunished weights of both alternatives and $p(B)$ as the sum of the punishments of both alternatives. That means, we solve the following optimization problem:

$$\min_{B \in S} f_\varepsilon(B)$$
$$\text{with } f_\varepsilon(B) = w(B) + \varepsilon \cdot p(B).$$

The following algorithm finds all threshold parameters. Let $\varepsilon_1 < \varepsilon_2 < \ldots < \varepsilon_k$ be these threshold parameters, then the algorithm also gives us alternatives which are optimal for the different optimality intervals. We denote the set $\mathcal{T}$ as the set of the threshold parameters that are found and $\mathcal{B}$ as the set of the generated alternatives.

(i) Generate B_0 and B_∞. By the definitions of our penalty methods, B_0 is a solution which minimizes $w(B)$ and has a $p(B)$ value as small as possible. Analogously, B_∞ minimizes $p(B)$ and has a $w(B)$ value which is as small as possible.

In case of $p(B_0) = p(B_\infty)$ the solution B_0 is optimal for all penalty parameters $\varepsilon \geqslant 0$ and thus no threshold parameter exists. We set $\mathcal{T} := \emptyset$ and $\mathcal{B} := \{B_0\}$. END.

(ii) Set the number of found alternatives k to two, because B_0 and B_∞ are different. Insert the investigated penalty parameters into a set E and insert the generated solutions into a set $\mathcal{B}$. Then, we also introduce Threshold-variables, which denote that a threshold parameter is found.
$k := 2$,
$\mathcal{T} := \emptyset$,
$E := (\varepsilon(1), \varepsilon(2)) = (0, \infty)$,
$\mathcal{B} := \{B_0, B_\infty\}$,
Threshold$(1) = false$.

(iii) Choose an $i \in \{1, 2, \ldots, k-1\}$ with Threshold$(i) = false$.

(*iv*) Calculate $\bar{\varepsilon}$ such that $f_{\bar{\varepsilon}}(B_{\varepsilon(i)}) = f_{\bar{\varepsilon}}(B_{\varepsilon(i+1)})$ holds:

$$w(B_{\varepsilon(i)}) + \bar{\varepsilon} \cdot p(B_{\varepsilon(i)}) = w(B_{\varepsilon(i+1)}) + \bar{\varepsilon} \cdot p(B_{\varepsilon(i+1)})$$
$$\Leftrightarrow \bar{\varepsilon} := \frac{w(B_{\varepsilon(i+1)}) - w(B_{\varepsilon(i)})}{p(B_{\varepsilon(i)}) - p(B_{\varepsilon(i+1)})}.$$

(*v*) Generate an optimal solution $B_{\bar{\varepsilon}}$ for the penalty parameter $\bar{\varepsilon}$.

(*vi*) If $f_{\bar{\varepsilon}}(B_{\bar{\varepsilon}}) = f_{\bar{\varepsilon}}(B_{\varepsilon(i)}) = f_{\bar{\varepsilon}}(B_{\varepsilon(i+1)})$ then $\bar{\varepsilon}$ is a threshold parameter and we set:
Threshold$(i) := true$,
$\mathcal{T} := \mathcal{T} \cup \bar{\varepsilon}$.

Otherwise, we found a new alternative and set:
$E := (\varepsilon(1), \ldots, \varepsilon(i), \bar{\varepsilon}, \varepsilon(i+1), \ldots, \varepsilon(k))$,
$\mathcal{B} := \mathcal{B} \cup B_{\bar{\varepsilon}}$,
Threshold:= (Threshold(1),...,Threshold(i), $false$,Threshold$(i+1)$,...,Threshold$(k-1)$),
$k := k + 1$.

(*vii*) If there exists any i with Threshold$(i) = false$ then GO TO STEP (*iii*), else END.

At the end of this algorithm $\mathcal{B}$ is a minimal sequence of different ε-optimal solutions such that $\mathcal{B}$ contains a penalty alternative for every $\varepsilon \geqslant 0$. The set $\mathcal{T}$ contains all threshold parameters of our problem. We can find all $|\mathcal{T}|$ threshold parameters of our problem by solving $2|\mathcal{T}| + 1$ optimization problems of the type $\underset{B \in S}{min} f_\varepsilon(B)$.

2.3 Valuated Matroids

In this section we introduce the basics of matroid theory that are directly needed for this thesis. Many more results for matroids can be found in the book by Welsh [Wel 1976]. One problem of the matroid theory is that there are many equivalent ways to define a matroid. Furthermore, many concepts within the matroid theory have a variety of equivalent formulations. In this thesis we use the formulation of Dress and Wenzel [DW 1990].

Althöfer and Wenzel [AW 1999a] already studied the problem of generating alternative solutions for valuated matroids. But they do not use the penalty methods to generate their alternatives but the "k-best under distance constraints" method. This method works as follows: They generate the optimal solution, then they generate the best solution which differs in at least one element from the optimal one, then the best solution which differs in at least two elements from the optimal solution and so on. Here, one nice result is that there exists a structural monotonicity. That means that in every new alternative at least one element of the preceding solution is

replaced by a new element. The replaced element is also part of the optimal solution and the new element is not part of any alternative for a smaller constraint. Then all succeeding solutions (which differ in still more elements from the optimal solution) use the new element and do not use the replaced one.

Definition 2.3.1 (Valuated Matroid).
Let E be a finite set. Assume $0 \leqslant m \leqslant |E|$ and let $v : \binom{E}{m} \to \mathbb{R} \cup \{-\infty\}$ be a weight function on the m-subsets of E. Then the pair $M_v = (E, v)$ is called ***valuated matroid*** *of* ***rank*** *m, if the following two axioms hold:*

$(V0)$ *There exists a* ***base*** *$B \in \binom{E}{m}$ with $v(B) \neq -\infty$.*

$(V1)$ *For $B_1, B_2 \in \binom{E}{m}$ and $e \in B_1 \setminus B_2$ there exists $f \in B_2 \setminus B_1$ with*

$$v(B_1) + v(B_2) \leqslant v((B_1 \setminus \{e\}) \cup \{f\}) + v((B_2 \setminus \{f\}) \cup \{e\}).$$

The set $\mathfrak{B}_v = \left\{B \in \binom{E}{m} : v(B) \neq -\infty\right\}$ is called the ***set of bases*** *of the valuated matroid M_v.*

Remark 2.3.2.
For the element $-\infty$ we use the following conventions:

- $-\infty + x = -\infty$ for all $x \in \mathbb{R}$
- $(-\infty) + (-\infty) = -\infty$,
- $-\infty \leqslant -\infty$.

As we want to investigate sum type problems, we specialize our definition of matroids. Therefore, we only have to modify the definition of the weight function.

Definition 2.3.3 (Valuated Matroid of Sum Type).
Assume $M = (E, \mathfrak{B}_v)$ is a matroid of rank m with the set of bases $\mathfrak{B}_v$ and let $v : E \to \mathbb{R}$ be a weight function on E. We define $v : \binom{E}{m} \to \mathbb{R} \cup \{-\infty\}$ as follows:

$$v(B) := \begin{cases} \sum_{e \in B} v(e), & \text{for } B \in \mathfrak{B}_v, \\ -\infty, & \text{otherwise.} \end{cases}$$

Then $M_v = (E, v)$ is a valuated matroid of sum type with the base set $\mathfrak{B}_v$.

Remark 2.3.4.
For valuated matroids of sum type, Axiom $(V1)$ from Definition 2.3.1 can be simplified as follows:

$(V1')$ For $B_1, B_2 \in \mathfrak{B}$ and $e \in B_1 \setminus B_2$ there exists $f \in B_2 \setminus B_1$ with

$$v(B_1) + v(B_2) = v((B_1 \setminus \{e\}) \cup \{f\}) + v((B_2 \setminus \{f\}) \cup \{e\}).$$

Definition 2.3.5 (Greedy Algorithm).
*The **greedy algorithm** is defined as follows:*

Step 0: *Choose any base* $B_0 = \{e_1, \ldots, e_m\} \in \binom{E}{m}$ *with* $v(B_0) \neq -\infty$

Step k: $(1 \leqslant k \leqslant m)$ *We assume that* $B_{k-1} = \{f_1, \ldots, f_{k-1}, e_k, \ldots, e_m\}$ *is already determined and choose* $f_k \in (E \setminus B_{k-1}) \cup \{e_k\}$ *with* $v(\{f_1, \ldots, f_k, e_{k+1}, \ldots, e_m\}) \geqslant v(\{f_1, \ldots, f_{k-1}, x, e_{k+1}, \ldots, e_m\})$ *for all* $x \in (E \setminus B_{k-1}) \cup \{e_k\}$.
Set $B_k := \{f_1, \ldots, f_k, e_{k+1}, \ldots, e_m\}$.

In [DW 1990] it was shown that the greedy algorithm generates an optimal solution for every starting base B_0. This reference gives some more results for this greedy algorithm and its relation to valuated matroids.

Remark 2.3.6.
The MST problem can be understood as a problem based on the structure of valuated matroids. We show this by transforming the MST problem to a valuated matroid:

- We have to transform the minimization problem MST to a maximization problem (valuated matroids were defined as maximization problems). Therefore we replace the weight function w by $-w$.
- Let n be the number of vertices in the given graph G. Then we set the rank of the matroid to $n - 1$.
- The set of bases of the matroid is the set of all spanning trees in G.

Remark 2.3.7.
In difference to the algorithms of Kruskal and Prim (see [CLR 1990]) the greedy algorithm from Definition 2.3.5 does not construct a solution (construction-greedy). It starts with a starting base B_0 and exchanges elements of this solution by better ones. Because of this exchange of elements, it is also called exchange-greedy. This algorithm works analogously for the MST problem. There, we start with an arbitrary spanning tree $B_0 = \{e_1, e_2, \ldots, e_{n-1}\}$ and perform the exchange step for every edge e_k, $1 \leqslant k \leqslant n - 1$ of B_0. In this way we get an MST.

2.4 The Minimum Cost Flow Problem and the Successive Shortest Path Algorithm

As the generation of mutual penalty alternatives with the help of linear programming turned out to be very time intensive, we have to find a better approach for their generation. It turns out that we can generate the alternatives for the shortest path problem and the assignment problem quite faster with the help of some results for the minimum cost flow problem. With this ulterior motive, we recapitulate the minimum cost flow problem. Then, we formulate the successive shortest path algorithm to find a minimum cost flow.

The basic network flow problem is the maximum flow problem. It is defined as follows:

Definition 2.4.1 (Maximum Flow Problem).
Let $G = (V, E)$ be a directed graph with edge set E and vertex set V. We denote $s \in V$ as source and $t \in V$ as sink. Every edge $(i, j) \in E$ has a nonnegative capacity $c(i, j)$. With $S(i) = \{j \in V : (i, j) \in E\}$ we denote the set of successors of i and with $P(i) = \{j \in V : (j, i) \in E\}$ the set of predecessors of i. A flow f is called feasible, if the following two conditions hold:

- $0 \leqslant f(i, j) \leqslant c(i, j)$ *for all* $(i, j) \in E$
- $\sum_{j \in S(i)} f(i, j) = \sum_{j \in P(i)} f(j, i)$ *for all* $i \in V \setminus \{s, t\}$ *(*KIRCHHOFF *conditions)*

The goal of the maximum flow problem is to find a feasible flow from s to t as large as possible, i.e. we want to maximize $\sum_{i \in P(t)} f(i, t)$.

A maximum flow can be found with the Ford-Fulkerson algorithm (see [AMO 1993, p. 180]). The basic concept of this algorithm is to augment the flow along $s-t$-paths as much as possible. After an augmenting step the capacities of the used edges are updated and backward edges are introduced to allow the reduction of the flow along these edges. The reduction of flows along several edges might be necessary to find other augmenting paths. Using these backward edges causes a reduction of the flow along its original edge, but it does not reduce the flow from s to t.

The minimum cost flow problem is a generalization of the maximum flow problem. Here, we have additional edge weights. The target is to find a maximum flow and amongst all of them a flow with minimum cost. We simplify the definition of the minimum cost flow problem and denote the maximum flow as $f_{max} = \sum_{i \in P(t)} f(i, t)$.

Definition 2.4.2 (Minimum Cost Flow Problem).
Let $G = (V, E)$ be a directed graph with edge set E and vertex set V. We denote $s \in V$ as source and $t \in V$ as sink. Every edge $(i, j) \in E$ has a nonnegative capacity $c(i, j)$ and a nonnegative weight per flow unit $w(i, j)$. With $S(i) = \{j \in V : (i, j) \in E\}$ we denote the set of successors of i and with $P(i) = \{j \in V : (j, i) \in E\}$ the set of predecessors of i. Let f_{max} be the maximum flow of G. Then the minimum cost flow problem is formulated as follows:

$$
\begin{aligned}
& min \sum_{e \in E} f(e) \cdot w(e) & & \\
s.t. \quad & 0 \leqslant f(i, j) \leqslant c(i, j) & & \text{for all } (i, j) \in E \\
& \sum_{i \in P(j)} f(i, j) = \sum_{j \in S(i)} f(i, j) & & \text{for all } i \in V \setminus \{i, j\} \text{ (KIRCHHOFF conditions)} \\
& \sum_{i \in P(t)} f(i, t) = f_{max}. & &
\end{aligned}
$$

One algorithm to find a minimum cost flow is the successive shortest path algorithm (see also [AMO 1993, pp. 320-324]). It is based on the algorithm of Ford and Fulkerson with the difference that it cares about the edge weights. Here, we formulate a modification of the basic successive shortest path algorithm which modifies the edge weights during the process. These modifications assure to have only nonnegative edge weights all over the process. Thus, we can use the Dijkstra algorithm to determine our shortest paths in G. As this algorithm and proofs of its correctness can be found in many books (for example [AMO 1993, pp. 320-324]), we do not give any proof for it in this thesis.

1. Set $f(x, y) = 0$ for all $(x, y) \in E$.

2. Determine the length $d(i)$ of the shortest path from s to i for all $i \in V$ with the Dijkstra algorithm. If there does not exist any path from s to t then STOP, f is the minimum cost flow.

3. Let c^* be the minimum capacity of the edges of the shortest $s-t$-path P. Then perform the following modifications:

   - $w(x, y) := w(x, y) + d(x) - d(y)$ for all $(x, y) \in E$,
   - $f(x, y) := f(x, y) + c^*$ for all forward edges $(x, y) \in P$,
   - $f(x, y) := f(x, y) - c^*$ for all backward edges $(y, x) \in P$,
   - $c(x, y) := c(x, y) - c^*$ for all $(x, y) \in P$.
     Delete all edges (x, y) which have a capacity $c(x, y) = 0$ after this modification.
   - For all edges $(x, y) \in P$ do:
     - If the backward edge (y, x) already exists, then set $c(y, x) := c(y, x) + c^*$,
     - Otherwise introduce the backward edge (y, x) and set $c(y, x) := c^*$ and $w(y, x) = 0$.

   GOTO step 2

With the help of this algorithm we will generate our mutual penalty alternatives for the shortest path problem and the assignment problem. On page 47 we also demonstrate this algorithm by generating mutual penalty alternatives for the shortest path problem.

Chapter 3

Simple Penalty Method

In this thesis we investigate the simple penalty method and the mutual penalty method. We start the investigation with the simple penalty method as it is easier to understand. The simple penalty method is used, for example, by the vehicle routing program [AND 1997] with $\varepsilon = 0.2$ to generate an alternative solution. Many other programs also use this approach.

In this chapter we determine the maximum number of threshold parameters for the shortest path problem and for valuated matroids. Regarding the shortest path problem we generalize the results of [Doe 2008]. There, directed acyclic graphs were investigated and an upper bound of $\frac{n^2}{2} - \frac{3n}{2} + 1$ was found. Here, we generalize these results to arbitrary multigraphs and get $n^2 - 5n + 10$ as an upper bound for the number of threshold parameters. By giving an example with $\frac{n^2}{4} - 1$ threshold parameters, we show that the maximum number of threshold parameters lies in the class $\Theta(n^2)$.

For valuated matroids there exists a structural monotonicity. This property gives us an upper bound of $rank(M)$ for the number of threshold parameters. To show that this bound is tight, we give an example for the MST problem which exactly meets this upper bound. Thus, the maximum number of threshold parameters is exactly $n - 1$ for the minimum spanning tree problem and $rank(M)$ for valuated matroids.

Regarding the assignment problem and the TSP we give examples with $\frac{n^2}{4} - 1$ and $\frac{(n-1)^2}{4} - 1$ threshold parameters. Thus, we get lower bounds of $\Omega(n^2)$ for the maximum number of threshold parameters.

3.1 Shortest Path Problem

Theorem 3.1.1.
Let G be an arbitrary directed graph with n vertices and let s be the start vertex and t the target vertex. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E. Let $P = (s = i_1, i_2, \dots i_{k-1}, i_k = t)$ be the shortest $s-t$-path that will be punished by the simple penalty method. Then the simple penalty method has at most $k^2 - 5k + 10$ threshold parameters for the shortest path problem.

Proof
If the shortest path from s to t in G is not unique, then we choose one of them and punish all of its edges to generate the simple penalty alternative. We denote the shortest path that is chosen as $P = (s = i_1, i_2, \dots i_{k-1}, i_k = t)$. Now we reduce the graph G in a way that the resulting graph $\tilde{G} = (\tilde{V}, \tilde{E})$ contains the vertices lying on P and no other vertex. We start the construction of $\tilde{G}$ with $\tilde{V} = \{s = i_1, i_2, \dots i_{k-1}, i_k = t\}$ and insert the punished edges $i_j \to i_{j+1}$, $1 \leqslant j \leqslant k-1$ into $\tilde{G}$. These edges get weight $(1+\varepsilon) \cdot w(i_j, i_{j+1})$. Now we remove all edges of P from G and generate the shortest (i, j) path for all pairs (i, j) with $i, j \in \tilde{V}$ in the graph $G \setminus P$. For every pair (i, j) we insert (if a shortest $i - j$-path in $G \setminus P$ exists) an edge (i, j) into $\tilde{G}$. The weight of this edge is the weight of the shortest $i - j$-path. Using this edge in any alternative means to use the shortest unpunished subpath from i to j. As paths with circuits cannot be optimal, we delete all edges going back to s or going away from t from our graph $\tilde{G}$.

We demonstrate the construction of $\tilde{G}$ with the following graph:

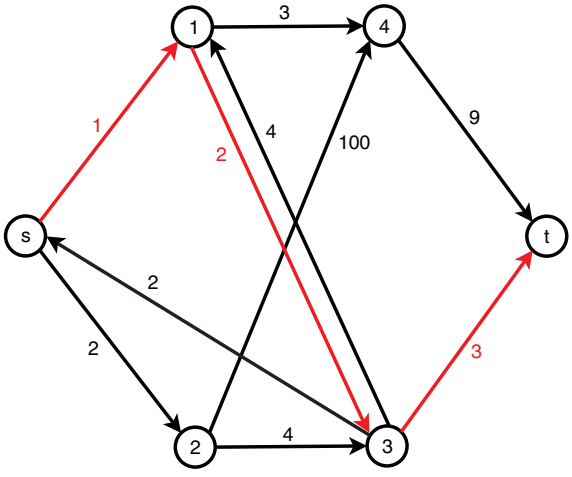
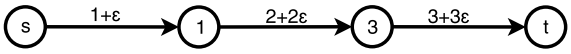


Figure 3.1: Construction of the Graph $\tilde{G}$ for the Simple Penalty Method, Basic Graph

The shortest path from s to t is the path $s \to 1 \to 3 \to t$. It is already printed with red color in Figure 3.1. Thus, the vertices s, 1, 3 and t

build the vertex set $\tilde{V}$. First, we insert the edges $s \to 1$, $1 \to 3$ and $3 \to t$ into $\tilde{G}$. They have weight $\tilde{w}(e) = (1 + \varepsilon) \cdot w(e)$, as they indicate the punished edges in G. After this first construction step, $\tilde{G}$ looks as follows:



Now we insert the unpunished edges which represent unpunished subpaths. Therefore we delete all edges of P from G and generate the shortest paths for all pairs of vertices in $\tilde{V}$. This graph $G \setminus P$ looks as follows:

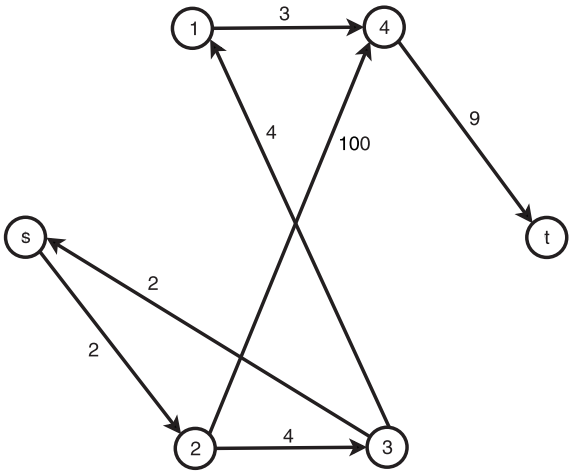


Figure 3.2: Construction of the Graph $\tilde{G}$ for the Simple Penalty Method, Determination of Shortest Unpunished Subpaths

Regarding the subpath $s \to 1$, the shortest path is $s \to 2 \to 3 \to 1$ with length 10. For the subpath $s \to 3$, we get the shortest path $s \to 2 \to 3$ with length 6 and so on. As the path $1 \to 3$ and paths starting in t or going back to s do not exist, $\tilde{G}$ will not contain the edge $(1, 3)$ and there does not exist any edge outgoing from $t$or going back to s. Altogether, $\tilde{G}$ looks as follows:

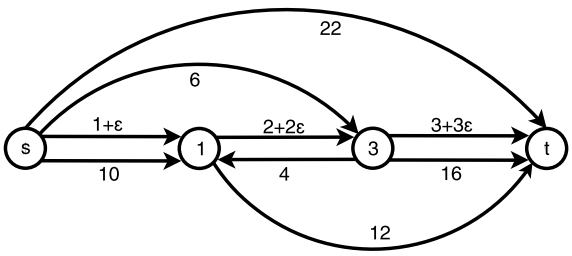


Figure 3.3: Completed Construction of the Graph $\tilde{G}$ for the Simple Penalty Method

In this graph we generate the shortest $s-t$-paths for every penalty parameter $\varepsilon \geqslant 0$. A "p" over an arrow means that the punished edge is used, otherwise the unpunished edge is used:

For $0 \leqslant \varepsilon \leqslant 1$ the path $s \xrightarrow{p} 1 \xrightarrow{p} 3 \xrightarrow{p} t$ is optimal.
For $1 \leqslant \varepsilon \leqslant 2$ the path $s \to 3 \xrightarrow{p} t$ is optimal.
For $2 \leqslant \varepsilon \leqslant 9$ the path $s \xrightarrow{p} 1 \to t$ is optimal.
And for $\varepsilon \geqslant 9$ the path $s \to t$ is optimal.

Translating these results for $\tilde{G}$ back to alternatives in G, we get:

For $0 \leqslant \varepsilon \leqslant 1$ the path $s \to 1 \to 3 \to t$ is optimal.
For $1 \leqslant \varepsilon \leqslant 2$ the path $s \to 2 \to 3 \to t$ is optimal.
For $2 \leqslant \varepsilon \leqslant 9$ the path $s \to 1 \to 4 \to t$ is optimal.
And for $\varepsilon \geqslant 9$ the path $s \to 2 \to 3 \to 1 \to 4 \to t$ is optimal.

Generating the shortest path in $\tilde{G}$ for a given parameter ε is equivalent to the generation of the simple penalty alternative for the same penalty parameter ε in G. For any fixed penalty parameter we get the same alternatives in G and in $\tilde{G}$. But the advantage of $\tilde{G}$ is to have an easy characterization of punished and unpunished edges.

Now, we use $\tilde{G}$ to find an upper bound for the maximum number of threshold parameters. First, we prove that new alternatives contain at least one new edge and then we use this property to formulate the upper bound.

For an easier characterization of our alternatives we rename the vertices in $\tilde{G}$ and enumerate them with $s = 1, 2, \ldots, k = t$ from the left to the right.

We claim that all alternatives do not contain two succeeding edges that are unpunished. If any alternative contains the edges (i, j) and (j, l) then the direct edge (i, l) is not longer, because it denotes the shortest unpunished path from i to l.

Now, we denote P_{new} as an alternative that is new at ε^*. With the help of the known properties for the penalty methods we compare P_{new} with alternatives which are optimal at $\varepsilon < \varepsilon^*$.

If P_{new} does not contain any interior vertex (vertices except s and t), then the direct edge (s, t) is used. The convexity property of Theorem 2.2.4 assures that this edge is new. If it was old, the alternative would also be optimal for a smaller penalty parameter as well for all penalty parameters in between, due to convexity. Thus, ε^* would not be a threshold parameter.

Thus, we assume that P_{new} uses some interior vertex i. Then, we investigate the subpaths $s \to i$ and $i \to t$ separately. Therefore we denote $\mathcal{P}_{old}$ as the set of alternatives that are optimal at $\varepsilon < \varepsilon^*$ and use the interior vertex i. As the monotonicity theorem (Theorem 2.2.3) also hold on subpaths, the punished weight

of the subpaths $s \to i$ and $i \to t$ decreases in ε. Let P_1 and P_2 be two alternatives in $\mathcal{P}_{old} \cup \{P_{new}\}$ and let P_2 be newer than P_1. Then, the punished weight of the subpaths $s \to i$ and $i \to t$ of P_2 cannot be larger than the punished weight of the corresponding subpaths of P_1. Furthermore, Remark 2.2.18 assures that the punishment of at least one of these subpaths is smaller than in P_1.

If P_{new} contains more than one interior vertex, then we find a subpath $i \to j$ at which the punishment is smaller than in all old alternatives, analogously.

To show that a new alternative contains at least one new edge, we investigate the following cases:

(i) The alternative P_{new} uses a new leftwards edge $(i, j), i > j$.

In this case we found the new edge (i, j).

(ii) The alternative P_{new} does not use any new leftwards edge.

We know that the punishment of at least one subpath of P_{new} is smaller than in all old alternatives. As all leftwards edges are unpunished (by construction) and old, the punishment cannot decrease at leftwards edges. (The alternative at which the leftwards edge was new uses the same leftwards subpath and thus the punishment cannot decrease here.) Thus, it remains to investigate rightwards subpaths. In at least one of them the punishment is smaller than in all old alternatives. We could use the include-exclude-algorithm as it was used in [Doe 2008], because in all rightwards subpaths leftwards edges are not allowed and thus we can generate these paths in DAGs. But here we use a shorter proof.

Let $s = i_0 \to i_1, i_2 \to i_3, \dots i_l \to i_{l+1} = t$ be these subpaths (this means that the leftwards edges $i_1 \to i_2, i_3 \to i_4, \dots i_{l-1} \to i_l$ are used). Then we compare these subpaths with the corresponding subpaths in old alternatives. Regarding the subpath $i_{2x} \to i_{2x+1}$, $0 \leqslant x < \frac{l}{2}$ we denote $\mathcal{P}_{old}$ as the set of alternatives that contain a subpath from i_{2x} to i_{2x+1}. It is assured that $\mathcal{P}_{old}$ is not empty, because $P_0 \in \mathcal{P}_{old}$. Amongst all of these alternatives in $\mathcal{P}_{old}$, we denote the newest one as P_{old}. If the punishment of the subpath $i_{2x} \to i_{2x+1}$ of P_{new} is smaller than the punishment of the subpath $i_{2x} \to i_{2x+1}$ of P_{old} then we investigate this subpath further in the following steps. Otherwise we investigate the next rightwards subpath of P_{new}.

Let $i_{2x} \to i_{2x+1}$ be a subpath with decreasing punishment. Then, we investigate the following two subpaths:

(a) The direct edge (i_{2x}, i_{2x+1}) is used by P_{new}.

Then, this edge is unpunished as only edges $i \to (i+1), 1 \leqslant i \leqslant k-1$ are punished and the punished length of P_{new} is smaller than the punished

length of all old alternatives. If there exists a punished edge $i_{2x} \to i_{2x+1}$ in $\tilde{G}$ then P_{new} also uses the unpunished edge, because the punishment decreases. Thus, this edge is new, otherwise the punishment would not decrease at this subpath.

(b) The alternative P_{new} uses the interior vertices $i_{j_1}, i_{j_2}, \ldots, i_{j_p}$ in this order and does not use any new leftwards edge.

We redefine $\mathcal{P}_{old}$ as the set of alternatives that are new at $\varepsilon < \varepsilon^*$ and use common interior vertices with the rightwards subpath $i_{2x} \to i_{2x+1}$ of P_{new}. Again, $\mathcal{P}_{old}$ is not empty, because $P_0 \in \mathcal{P}_{old}$.

Now, we denote P_{old} as the newest alternative amongst all alternatives of $\mathcal{P}_{old}$.

Let $i_{2x} = i_{m_1}, i_{m_2}, \ldots, i_{m_q} = i_{2x+1}$ be the vertices that are used by the alternatives P_{new} and P_{old} in this order. As the punishment of P_{new} decreases at ε^*, it also decreases in at least one of the subpaths $i_{m_y} \to i_{m_{y+1}}$, $0 < y < q$. We investigate one of these subpaths $i_{m_y} \to i_{m_{y+1}}$ with decreasing punishment recursively. This means, we set $i_{2x} := i_{m_y}$ and $i_{2x+1} := i_{m_{y+1}}$ and investigate the cases (a) and (b) for this subpath $i_{2x} \to i_{2x+1}$. After at most $n-1$ recursions we will come to case (a) and get a pair of vertices (i, j) with the property that the path from i to j used punished edges so far and is unpunished now. The convexity properties of Theorem 2.2.4 assure that this edge is new. If an alternative used this unpunished edge before and P_{new} uses it, then P_{old} also had to use it. But this contradicts our choice of (i, j). Thus, the direct edge (i, j) is used by P_{new} and it is new at ε^*.

As every new alternative contains a new edge (i, j) and $\tilde{G}$ does not contain edges going back to s or going away from t, we just have to count the unpunished edges in $\tilde{G}$ to find an upper bound for the number of threshold parameters:

- The vertex $s = 1$ has at most $(k-1)$ outgoing edges (to $2, 3, \ldots, k$).
- The vertices $2, 3, \ldots (k-1)$ have at most $(k-2)$ outgoing edges, because they have no outgoing edge to s and no edge to themselves.
- The vertex $t = k$ has no outgoing edge.

Altogether, there exist at most $(k-1) + (k-2) \cdot (k-2) = k^2 - 3k + 3$ unpunished edges in $\tilde{G}$.

We claimed that all alternatives do not contain two succeeding edges that are unpunished. If the leftwards edge $(i+1) \to i$ is used, then the alternative uses a punished edge, afterwards. The only punished edge that is going out of i is the edge $i \to (i+1)$. Thus, we get a positive circuit $(i+1) \to i \to (i+1)$ and the alternative cannot be optimal.

Analogously, we rule out edges $(i+2) \to i$, because the edge $i \to (i+1)$ is used afterwards and the punished edge $(i+1) \to (i+2)$ is used before $(i+2) \to i$, because it is the only punished edge going to $i+2$. Thus, we get the positive circuit $(i+1) \to (i+2) \to i \to (i+1)$.

As these leftwards edges cannot be used by our alternative, we can reduce $\tilde{G}$ by deleting all $k-3$ edges of the kind $i \to (i-1)$ and all $k-4$ edges of the kind $i \to (i-2)$. After this reduction, $\tilde{G}$ contains $k^2-3k+3-(k-3)-(k-4) = k^2-5k+10$ edges.

As every new alternative contains a new edge in $\tilde{G}$, we get the upper bound of $k^2-5k+10$ for the number of threshold parameters. ■

Because of $k \leqslant n$, we especially found an upper bound of $n^2-5n+10$ for the maximum number of threshold parameters.

In this proof we did not use the property that only one edge (i,j) exists in G. We analogously can transform an arbitrary multigraph to our simplified graph $\tilde{G}$. As Dijkstra's algorithm works analogously in the case of multigraphs, nothing changes. Thus, our upper bound for graphs without multiple edges also holds for multigraphs.

It remains to give a lower bound for the maximum number of threshold parameters. Here, [Doe 2008] already gave a lower bound of $\frac{n^2}{4}-1$. This example can be found in Appendix A.1 of this thesis. With this lower bound and our upper bound of $n^2-5n+10$, the maximum number of threshold parameters lies in the class $\Theta(n^2)$.

3.2 Valuated Matroids

Applying the simple penalty method to valuated matroids, we get a structural monotonicity. This means that at every threshold parameter an element e of the optimal solution is replaced by a new element f. For all larger penalty parameters the element e will not be part of the penalty alternative and f will be part of all of these alternatives.

Theorem 3.2.1 ([Doe 2008]).
Let E be a finite set. Assume $0 \leqslant m \leqslant |E|$ and let $v : E \to \mathbb{R}^-$ be an injective weight function on E. Let the pair $M_v = (E, v)$ be a valuated matroid of sum type with $rank(M_v) = m$. Then the following two properties hold:

(i) For all $\delta < \varepsilon$ and for all $a \in B_0 \setminus B_\delta$ holds $a \notin B_\varepsilon$.

(ii) For all $\delta < \varepsilon$ and for all $b \in B_\delta \setminus B_0$ holds $b \in B_\varepsilon$.

This theorem was part of the diploma thesis [Doe 2008]. Its proof can also be found in Appendix A.2 of this thesis.

Theorem 3.2.2.
Let E be a finite set. Assume $0 \leqslant m \leqslant |E|$ and let $v : E \to \mathbb{R}^-$ be an injective weight function on E. Let the pair $M_v = (E, v)$ be a valuated matroid of sum type with $rank(M_v) = m$. If the weight function v is injective then the optimal base is unique.

Proof
Let v be an injective weight function and $\xi > 0$ with $\min\limits_{x,y \in E} |v(x) - v(y)| \geqslant \xi$. We assume that there exist two optimal bases B_1 and B_2 with $B_1 \neq B_2$.

The base B_1 is locally optimal, this means that there is no exchange of elements $e \in B_1$ and $f \notin B_1$ possible in a way that $(B_1 \setminus \{e\}) \cup \{f\}$ is a base and has a smaller weight than B_1. If there was any local improvement possible, the new base would have a weight of at least $v(B_1) + \xi$.

Because of $B_1 \neq B_2$, there exists an element e with $e \in B_1 \setminus B_2$. We modify its weight to $\tilde{v}(e) := v(e) - \frac{\xi}{2}$, all other weights remain unmodified.

Let $\tilde{B_1}$ be the base B_1 with the modified weight of e. The greedy algorithm starting in $\tilde{B_1}$ cannot improve the weight of the starting base $\tilde{B_1}$ by any exchange because every element which was larger than e before the modification is still larger than e and every element which was smaller is still smaller. Thus, it does not make any sense to exchange e with any other element. Analogously no other element of $\tilde{B_1}$ can be replaced by a better one. So the greedy algorithm again gives us $\tilde{B_1}$ as the optimal base even though B_2 is better, because its weight was not modified. As the greedy algorithm always generates an optimal solution, the other optimal base B_2 cannot exist. ■

Corollary 3.2.3.
Let E be a finite set. Assume $0 \leqslant m \leqslant |E|$ and let $v : E \to \mathbb{R}^-$ be an injective weight function on E. Let the pair $M_v = (E, v)$ be a valuated matroid of sum type with $rank(M_v) = m$. Then there are at most m threshold parameters possible.

Proof
Theorem 3.2.1 gives us a structural monotonicity. This means that at every threshold parameter an element which is also part of the base B_0 falls out of the generated alternative and will not be part of any alternative for a larger penalty parameter. Furthermore, every element which is new at any threshold parameter will be part of every ε-optimal alternative for all larger penalty parameters ε. As the base B_0 contains m elements, we can get at most m threshold parameters. ■

Remark 3.2.4.
The injectivity of the weight function assures the uniqueness of the optimal solution. If the weight function is not injective then we add very small extra weights to edges which have equal weights. If the weight of an element e is smaller than the weight of another element f, then it will also be smaller after this modification. This modification can be done by the following scheme analogously to [AW 1999a].

We choose any $\xi > 0$ with $\xi < v(e) - v(f)$ for all $e, f \in E$ with $v(e) > v(f)$. Now we use a fixed permutation of the elements of E: $(e_1, e_2, \ldots, e_{N-1}, e_N)$ with $N = |E|$. For all elements e_i we define:

$$\tilde{v}\,(e) = v\,(e) + 10^{-i} \cdot \xi$$

With this scheme we get an injective weight function $\tilde{v}$ on E. If the generated alternative for the problem with modified weights is worse (in respect to the original weights) than the alternative in the original problem for the same penalty parameter, then we denote the difference with Δ. Choosing $\xi < \frac{\Delta}{N \cdot (1+\varepsilon)}$ under retention of the other claims for ξ solves this problem. Thus, we can choose our $\xi > 0$ in a way that the set of alternatives in the modified problem also gives us an optimal alternative for every penalty parameter ε in the original problem.

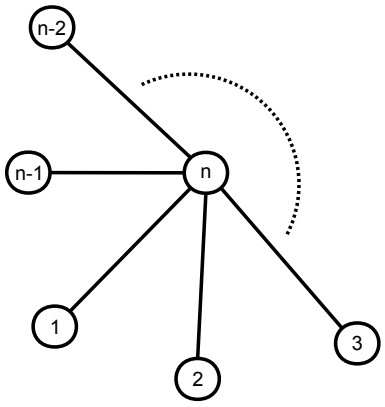
Example 3.2.5.
There exist examples which meet this bound of $m = rank\,(M_v)$ threshold parameters. We demonstrate this fact with the MST problem by giving a graph with n vertices. Here the rank of the matroid is $n - 1$. Let $1, 2, 3, \ldots, n$ denote the different vertices. Then we want to get the following sequence of alternatives:

1. The optimal solution is the line $1 - 2 - 3 - \ldots - n$, that means it contains the edges $\{i, i+1\}$ for $i < n$.

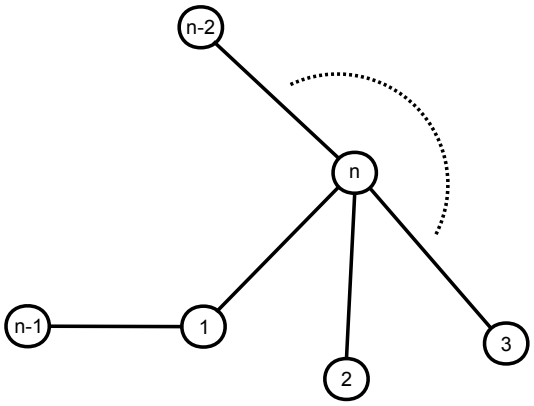
2. The edge $\{1, 2\}$ is replaced by $\{1, n\}$ at the threshold parameter $\varepsilon = n - 1$.
3. The edge $\{2, 3\}$ is replaced by $\{2, n\}$ at the threshold parameter $\varepsilon = n$.

$\vdots$

(n-2). The edge $\{n-2, n-1\}$ is replaced by $\{n-2, n\}$ at the threshold parameter $\varepsilon = 2n - 4$. Now the spanning tree is a star with center n.



(n-1). The only remaining edge of the optimal solution is $\{n-1, n\}$. It is replaced by $\{n-1, 1\}$ at the threshold parameter $\varepsilon = 2n-3$.



As edge weights of the optimal solution we use $w(i, i+1) = i$ for $0 < i < n$. The weights of the other edges can be calculated by setting $w(i, n) = w(i, i+1) \cdot (1 + \varepsilon_i)$ with $0 < i < n-1$ and the threshold parameter ε_i for which $(i, i+1)$ is replaced by (i, n). Analogously, we set $w(n-1, 1) = (n-1) \cdot (2n-2)$. Altogether, we get:

$$w(i,j) = \begin{cases} i, & \text{for } i < n,\ j = i+1 \\ i \cdot (n+i-1), & \text{for } i < n-1,\ j = n \\ 2 \cdot (n-1)^2, & \text{for } i = 1,\ j = n-1. \end{cases}$$

Other edges do not exist in our graph G.

3.3 Assignment Problem

In this section we construct a scenario with $\frac{n^2}{4} - 1$ threshold parameters for the assignment problem. First, we demonstrate our construction for the case $n = 6$ and then we generalize it to an arbitrary dimension n.

Example 3.3.1.
In this example the weights are chosen such that all threshold parameters and weights are natural numbers. Regarding the sequence of alternatives we use the following scheme:

1. At first, agent i performs task i for all i.

2. Then, agent 1 changes his task and performs task 2. Thus, agent 2 has to perform task 1 to repair the assignment. Then agent 1 wants to perform task 3. So agent 2 can perform his favorite task 2 again, but agent 3 has to repair the assignment by performing task 1. Step by step agent 1 performs all tasks from 1 to n and the agent who performed this task before has to perform task 1. For all following alternatives agent 1 performs task n and agent n performs task 1.

3. Now, agent 2 starts to change his tasks analogously with the agents $3, 4, \ldots, n-1$. Then for all following alternatives agent 2 performs task $n-1$ and agent $n-1$ performs task 2.

4. Then, agent 3 starts to change his tasks and so on.

Altogether, we want to get the alternatives which are denoted in the following table:

$\varepsilon \in$	Assignment					
$[0,1]$	(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)
$[1,2]$	**(1,2)**	**(2,1)**	(3,3)	(4,4)	(5,5)	(6,6)
$[2,3]$	**(1,3)**	(2,2)	**(3,1)**	(4,4)	(5,5)	(6,6)
$[3,4]$	**(1,4)**	(2,2)	(3,3)	**(4,1)**	(5,5)	(6,6)
$[4,5]$	**(1,5)**	(2,2)	(3,3)	(4,4)	**(5,1)**	(6,6)
$[5,6]$	**(1,6)**	(2,2)	(3,3)	(4,4)	(5,5)	**(6,1)**
$[6,7]$	*(1,6)*	**(2,3)**	**(3,2)**	(4,4)	(5,5)	*(6,1)*
$[7,8]$	*(1,6)*	**(2,4)**	(3,3)	**(4,2)**	(5,5)	*(6,1)*
$[8,9]$	*(1,6)*	**(2,5)**	(3,3)	(4,4)	**(5,2)**	*(6,1)*
$[9,\infty]$	*(1,6)*	*(2,5)*	**(3,4)**	**(4,3)**	*(5,2)*	*(6,1)*

Table 3.1: All 10 Alternative Assignments of our Example

We set the weights of all elements in the optimal solution to $c(i,i) = 2i$. As every new alternative in Table 3.1 uses exactly two new cells, we give both of them the same weight. In this way the calculation of the remaining weights is quite easy:

1. Choose ε as the smallest threshold parameter which was not used so far.
2. Calculate the weights of the two new cells such that both get the same weight and the alternatives which are optimal have the same weight for the threshold parameter ε.

So the weights of the cells $(1,2)$ and $(2,1)$ are calculated as follows:

$$\begin{aligned}
&(1+\varepsilon)\cdot[c(1,1)+c(2,2)+c(3,3)+c(4,4)+c(5,5)+c(6,6)] \\
&\qquad = (1+\varepsilon)\cdot[c(3,3)+c(4,4)+c(5,5)+c(6,6)]+c(1,2)+c(2,1) \\
\Leftrightarrow\quad &(1+1)\cdot(2+4+6+8+10+12) = (1+1)\cdot(6+8+10+12)+c(1,2)+c(2,1) \\
\Leftrightarrow\quad &2\cdot 6 = c(1,2)+c(2,1)\,.
\end{aligned}$$

With $c(1,2) = c(2,1)$ we get $c(1,2) = c(2,1) = 6$. Choosing $\varepsilon = 2$, we get the following equations:

$$\begin{aligned}
&(1+\varepsilon)\cdot[c(3,3)+c(4,4)+c(5,5)+c(6,6)]+c(1,2)+c(2,1) \\
&\qquad = (1+\varepsilon)\cdot[c(2,2)+c(4,4)+c(5,5)+c(6,6)]+c(1,3)+c(3,1) \\
\Leftrightarrow\quad &(1+2)\cdot(6+8+10+12)+6+6 = (1+2)\cdot(4+8+10+12)+c(1,3)+c(3,1) \\
\Leftrightarrow\quad &18 = c(1,3)+c(3,1)
\end{aligned}$$

With $c(1,3) = c(3,1)$ we get $c(1,3) = c(3,1) = 9$. In the same way all other weights of cells which are used by any alternative can be calculated. By setting the weights of all cells which are not used by any alternative to ∞, the following weight matrix comes up:

	T_1	T_2	T_3	T_4	T_5	T_6
A_1	2	6	9	13	18	24
A_2	6	4	35	43	52	∞
A_3	9	35	6	70	∞	∞
A_4	13	43	70	8	∞	∞
A_5	18	52	∞	∞	10	∞
A_6	24	∞	∞	∞	∞	12

TABLE 3.2: The Cost Matrix $C(i,j)$ for an Assignment Problem with 9 Threshold Parameters

This example can be generalized to problems of dimension $n \times n$. As the equation $c(i,j) = c(j,i)$ holds for all i and j due to our construction, we give the formulas for the case $i \leqslant j$. The remaining weights can be set by switching the roles of i and j in the formulas.

$$c(i,j) = \begin{cases} 2i, & \text{if } i = j \\ \infty, & \text{if } i < j \text{ and } j + i > n + 1 \\ (i+j) \cdot \left[(i-1) \cdot n - (i-1)^2 + 2 \right] + \frac{(j-i-1) \cdot (j-i)}{2}, & \text{otherwise.} \end{cases}$$

3.4 Directed Traveling Salesperson Problem

Analogously to the shortest path problem and the assignment problem we show that there exist examples with $\Omega(n^2)$ threshold parameters for the directed TSP. This will be shown with an example with $\frac{(n-1)^2}{4}$ threshold parameters. We demonstrate this construction for $n = 7$ and generalize it for an arbitrary n, afterwards.

Example 3.4.1.
As it is quite difficult to verify the optimality of solutions for the TSP, we build our graph in a way that it is easy to see that the solutions are optimal in the proposed sequence:

1. Let the optimal tour be $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 1$.
2. First, vertex 7 changes its position in the tour and the order of all other vertices remains unmodified. In this way, vertex 7 is visited between the vertices 5 and 6 in the next alternative, then between 4 and 5 and so on, until it is between 1 and 2. There it remains in all following alternatives.
3. After that, vertex 6 starts its "walk". First, it is visited between 4 and 5, then between 3 and 4 and then between 2 and 3. There it remains in all following alternatives.
4. Then, vertex 5 starts its "walk"....

With this scheme, we get the alternatives which are denoted in the following table:

$\varepsilon \in$	Tour
$[0, \varepsilon_1]$	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 1$
$[\varepsilon_1, \varepsilon_2]$	$1 \to 2 \to 3 \to 4 \to 5 \to \mathbf{7} \to 6 \to 1$
$[\varepsilon_2, \varepsilon_3]$	$1 \to 2 \to 3 \to 4 \to \mathbf{7} \to 5 \to 6 \to 1$
$[\varepsilon_3, \varepsilon_4]$	$1 \to 2 \to 3 \to \mathbf{7} \to 4 \to 5 \to 6 \to 1$
$[\varepsilon_4, \varepsilon_5]$	$1 \to 2 \to \mathbf{7} \to 3 \to 4 \to 5 \to 6 \to 1$
$[\varepsilon_5, \varepsilon_6]$	$1 \to \mathbf{7} \to 2 \to 3 \to 4 \to 5 \to 6 \to 1$
$[\varepsilon_6, \varepsilon_7]$	$1 \to 7 \to 2 \to 3 \to 4 \to \mathbf{6} \to 5 \to 1$
$[\varepsilon_7, \varepsilon_8]$	$1 \to 7 \to 2 \to 3 \to \mathbf{6} \to 4 \to 5 \to 1$
$[\varepsilon_8, \varepsilon_9]$	$1 \to 7 \to 2 \to \mathbf{6} \to 3 \to 4 \to 5 \to 1$
$[\varepsilon_9, \infty]$	$1 \to 7 \to 2 \to 6 \to 3 \to \mathbf{5} \to 4 \to 1$

TABLE 3.3: All 10 Alternative Tours of our Example for the TSP ($n = 7$)

We choose $w(i, i+1) = 2^{(n-i)}$ for $1 \leqslant i \leqslant n-1$ and $w(n-1, n) = 2^0 = 1$ as weights of the optimal solution. Then, we calculate the weights of the other edges as follows:

1. The weights of the optimal solution are already known.
2. The second solution uses three new edges. To assure that the optimal solution remains optimal, we give these three edges weights which are larger than the length of the optimal tour. As the length of the optimal tour is $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 = 2^7 - 1$, we set the weights of the new edges to 2^7.
3. Then, we iterate as follows:
   (*i*) Choose the next alternative in Table 3.3.
   (*ii*) Let the previous inserted edge have weight 2^x. If an edge (i, j) is used by the new alternative and has no weight so far, then its weight is set to 2^{x+2}.

Thus, we have an easy method for the determination of the weights of all edges. It remains to calculate the threshold parameters.

The first threshold parameter is ε_1. Here the edges $(5, 7)$, $(7, 6)$ and $(6, 1)$ are used for the first time. They get weight 2^7 (see Step 2 above). Thus, the threshold parameter ε_1 can be determined as follows:

$$\begin{aligned}(1+\varepsilon) \cdot [w(1,2) + w(2,3) + w(3,4) + w(4,5) + w(5,6) + w(6,7) + w(7,1)] \\ = (1+\varepsilon) \cdot [w(1,2) + w(2,3) + w(3,4) + w(4,5)] + w(5,7) + w(7,6) + w(6,1)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \quad (1+\varepsilon_1) \cdot (64+32+16+8+4+2+1) &= (1+\varepsilon_1) \cdot (64+32+16+8) + 3 \cdot 128 \\ \Leftrightarrow \quad \varepsilon_1 &= \frac{377}{7} \approx 53.86\end{aligned}$$

Analogously, we get $\varepsilon_2 = 191$, $\varepsilon_3 = 383$, $\varepsilon_4 = 767$ and $\varepsilon_5 = 1535$.

Now, we already know some weights of our TSP instance and so we can fill some cells of our weight matrix. In the following weight matrix a "$-$" denotes that this edge does not exist and a blank cell means that this weight is not calculated so far. The meaning of the "X"-signs will be explained below, so far these cells can be understood as empty.

	1	2	3	4	5	6	7
1	$-$	2^6	X	X	X	X	2^{15}
2		$-$	2^5				2^{13}
3		X	$-$	2^4			2^{11}
4		X		$-$	2^3		2^9
5		X			$-$	2^2	2^7
6	2^7	X				$-$	2^1
7	2^0	2^{15}	2^{13}	2^{11}	2^9	2^7	$-$

TABLE 3.4: Weight Function for an Example with 9 Threshold Parameters for the TSP ($n = 7$), First Weights

As all new edges for any alternative have a larger weight than the sum of all edge weights of the alternative before, it cannot happen that these edges are used earlier than wanted.

To assure that vertex 7 remains between the vertices 1 and 2, we forbid all edges outgoing from 1 which were not used so far and all edges going to 2 which were not used so far. In this way we get the "X"-signs in Table 3.4.

With this restriction, there exist only two possible paths from 1 to 2: $1 \to 2$ and $1 \to 7 \to 2$. Choosing the edge $1 \to 2$ in any alternative for a larger penalty parameter contradicts the monotonicity property of the penalty methods (Theorem 2.2.3), because the punished weight of the new solution would increase by at least one. In this way Theorem 2.2.3 assures that all following alternatives use the subpath $1 \to 7 \to 2$. Thus, we build the supervertex 172 denoting the subpath $1 \to 7 \to 2$. With this modification the following reduced table comes up:

	172	3	4	5	6
172	–	2^5			
3		–	2^4		
4			–	2^3	
5				–	2^2
6	2^7				–

TABLE 3.5: Weight Function for an Example with 9 Threshold Parameters for the TSP ($n = 7$), Introduction of the Supervertex 172

Continuing the construction scheme, we get $\varepsilon_6 = 32756$, $\varepsilon_7 = 98303$ and $\varepsilon_8 = 196607$.

Analogously to the last step, we forbid all other edges outgoing from 2 and ingoing to 3 and get the following table:

	172	3	4	5	6
172	–	2^5	–	–	2^{21}
3		–	2^4		2^{19}
4		–	–	2^3	2^{17}
5	2^{17}	–		–	2^2
6	2^7	2^{21}	2^{19}	2^{17}	–

TABLE 3.6: Weight Function for an Example with 9 Threshold Parameters for the TSP ($n = 7$), Supervertex 172 and forbidden Edges

From this point on all alternatives do not only use the subpath $1 \to 7 \to 2$, but also the subpath $2 \to 6 \to 3$. Choosing the edge $2 \to 3$ in any alternative for a larger penalty parameter would contradict the monotonicity property of the penalty methods again. Thus, all following alternatives use the subpath $1 \to 7 \to 2 \to 6 \to 3$. By building the supervertex 17263 we get the following reduced table:

	17263	4	5
17263	–	2^4	
4		–	2^3
5	2^{17}		–

TABLE 3.7: Weight Function for an Example with 9 Threshold Parameters for the TSP ($n = 7$), Introduction of the Supervertex 17263

Now there is only one step remaining in order to get the last edge weights and the value of the last threshold parameter $\varepsilon_9 = 1043113.\bar{6}$. Thus, we know all weights of our graph and complete our weight table:

	1	2	3	4	5	6	7
1	–	2^6	–	–	–	–	2^{15}
2	–	–	2^5	–	–	2^{21}	2^{13}
3	–	–	–	2^4	2^{23}	2^{19}	2^{11}
4	2^{23}	–	–	–	2^3	2^{17}	2^9
5	2^{17}	–	–	2^{23}	–	2^2	2^7
6	2^7	–	2^{21}	2^{19}	2^{17}	–	2^1
7	2^0	2^{15}	2^{13}	2^{11}	2^9	2^7	–

TABLE 3.8: Completed Weight Function for an TSP Instance with 9 Threshold Parameters ($n = 7$)

This construction shows that the sequence of alternatives is indeed optimal for the different intervals of penalty parameters. Other alternatives cannot be optimal for any penalty parameter ε. It is also assured that the threshold parameters increase in this sequence of alternatives. Here, we have $\varepsilon_{i+1} \geqslant 2\varepsilon_i$.

With the given construction scheme we can construct graphs with $\frac{(n-1)^2}{4}$ threshold parameters for any number of vertices n and get the following weight function:

$$w(i,j) = \begin{cases} 2^{n-i}, & \text{for } j = i+1 \text{ or } (i,j) = (n,1), \\ 2^{2in-2i^2-2i+n}, & \text{for } j = 1, \left\lfloor \frac{n}{2} \right\rfloor + 1 \leqslant i \leqslant n-1, \\ 2^{2in-n-2i^2+4i-2j-2}, & \text{for } i,j \text{ with } n-i+1 < j < i < n, \\ 2^{2jn-n-2j^2+4j-2i-4}, & \text{for } i,j \text{ with } n-j < i < j < n-1, \\ \infty, & \text{otherwise.} \end{cases}$$

Chapter 4

Mutual Penalty Method

The simple penalty method is well understood, but it is difficult to see what the mutual penalty method does. The biggest problem is that we do not generate one alternative after the other one, but both alternatives simultaneously. In this chapter we prove some properties of the generated alternatives to make clearer what the mutual penalty method does.

Sameith [Sam 2005, p.40] showed that the mutual penalty alternatives are as good as the alternatives generated by the simple penalty method. But he preferred the simple penalty method as it was easier and faster to generate the alternatives. Thus, one goal was to find fast algorithms for the generation of mutual penalty alternatives. For the shortest path problem and the assignment problem we transform the problem of finding mutual penalty alternatives to a minimum cost flow problem and generate our pair of alternatives with the successive shortest path algorithm. For valuated matroids we use a modified greedy algorithm.

Regarding the shortest path problem we give three properties of the generated pairs of mutual penalty alternatives. If both alternatives use common vertices then these vertices are used in the same order in both alternatives. The second property shows that all common edges of the pairs of alternatives are also part of the shortest path. The third property gives the result that the alternatives use the vertices of the shortest path P_0 in the same order as they are used by P_0. With the help of these properties we find an upper bound for the number of threshold parameters. Therefore, we reduce the mutual shortest path problem on our graph G to a special parametric shortest path problem on a graph $\tilde{G}$. After this reduction the proof of the upper bound works similar to the proof of the upper bound for the simple penalty method. In this way we show that at most $\frac{n^2}{2} - \frac{n}{2}$ threshold parameters are possible. After that, we show that the maximum number of threshold parameters lies in the class $\Theta(n^2)$ by giving an example with $\frac{n^2}{4} - 1$ threshold parameters.

Regarding the valuated matroids we show that elements which are part of both alternatives are also part of the optimal base. In contrast to the shortest path method, it even holds that every element of the optimal base is part of at least one of the mutual penalty alternatives. Analogously to the simple penalty method and to [AW 1999a], we prove a structural monotonicity. With these properties we get the result that at most $rank(M)$ threshold parameters are possible. We show that this bound is tight by giving an example for the MST problem which exactly meets this upper bound. As all of these properties also hold for the simple penalty method, this might lead us to the assumption that the simple penalty method and the mutual penalty method generate the same alternatives. But we can disprove this assumption with the help of a counterexample for the minimum spanning tree problem.

Finally, we show for the assignment problem and the TSP that common elements of both mutual penalty alternatives do not have to be part of the optimal solution.

4.1 Shortest Path Problem

4.1.1 Three Basic Properties of Mutual Shortest Paths

Theorem 4.1.1.
Let G be an arbitrary directed graph with n vertices and let s be the start vertex and t the target vertex. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E. If both generated mutual penalty alternatives use two common vertices i and j, then they will use either i before j in both alternatives or j before i in both alternatives.

Proof
We assume that there exists a pair of alternatives P_1 and P_2 with the property that P_1 uses i before j and P_2 uses j before i.

By cutting both paths in the vertices i and j we get the subpaths $P_1(s,i)$, $P_1(i,j)$, $P_1(j,t)$, $P_2(s,j)$, $P_2(j,i)$ and $P_2(i,t)$. All these subpaths have a positive weight. Setting $P_3 = P_1(s,i) \cup P_2(i,t)$ and $P_4 = P_2(s,j) \cup P_1(j,t)$ gives us two paths from s to t by using only edges which are part of P_1 or P_2. As no edge of P_3 or P_4 can be punished which was not punished in P_1 or P_2, the following inequality holds:

$$w(P_3) + w(P_4) \leqslant w(P_1) + w(P_2) - w(P_1(i,j)) - w(P_2(j,i)).$$

As the weights $w(P_1(i,j))$ and $w(P_2(j,i))$ are positive, we have:

$$w(P_3) + w(P_4) < w(P_1) + w(P_2).$$

Thus, the paths P_1 and P_2 cannot be optimal and our assumption is wrong. ■

Theorem 4.1.2.
Let $G = (V, E)$ be an arbitrary directed graph and let s be the start vertex and t the target vertex. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E and let B_0 be the unique shortest path from s to t. Then no edge which is not part of B_0 can be used twice in any pair of alternatives.

Proof
We assume that an edge (i, j) which is not part of the optimal solution B_0 is used twice. We draw this graph G as follows:

- All vertices which are used by the shortest path are drawn on a imaginary line from the left to the right in the same order as they are used by the shortest path. Enumerate these vertices increasingly from the left to the right.
- All other vertices are drawn anywhere below this line.
- Both paths use the edge (i, j). Mark all vertices of the shortest path with blue color if they lie on a path from s to i and with red color if they lie on a path from j to t.

Theorem 4.1.1 assures that no vertex is is both red and blue.

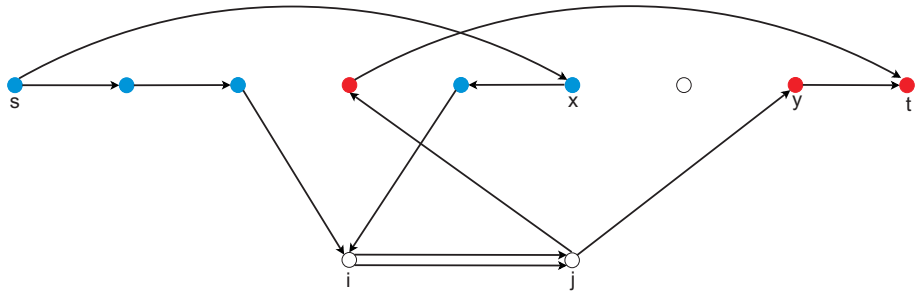


FIGURE 4.1: Proof of Theorem 4.1.2, Assumption that (i, j) is Used Twice

Now we choose a pair (x, y) with $x < y$ in a way that x is blue, y is red and no other colored vertex lies between them. Such a pair must exist because s is blue and t red. Choosing the blue vertex with the largest index as x, then there exists at least one red vertex with a larger index – the vertex t. We choose the red vertex as y which is to the right of x and has the smallest index amongst all of them. In Figure 4.1 these vertices are already labeled with x and y.

The edges leading from x to y which were used by the shortest $s - t$-path are not used by any of our two alternative paths. Otherwise at least one vertex between x and y would be colored or x and y had the same color. We denote $X \to Y$ as the path from x to y as it was used in B_0 and $w(X \to Y)$ as its weight. As B_0 is the optimal solution we get $w(X \to Y) < w(x \to \cdots \to i \to j \to \cdots \to y)$.

So the alternative using (i, j) cannot be optimal, because it would be better to choose the subpath $X \to Y$ instead of the path $x \to \cdots \to i \to j \to \cdots \to y$ which was assumed to be optimal. As no edge of $X \to Y$ is used so far, we do not have to care about punishments. Thus, the following pair of alternatives is better than P_1 and P_2 and our assumption is wrong.

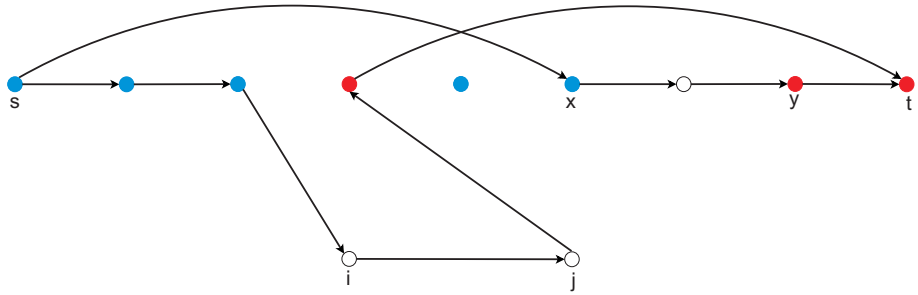


FIGURE 4.2: Proof of Theorem 4.1.2, Modified Paths

■

Theorem 4.1.3.
Let $G = (V, E)$ be an arbitrary directed graph and let s be the start vertex and t the target vertex. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E and let B_0 be the unique shortest path from s to t. Then all vertices which are used by B_0 are used in the same order (if they are used) by the mutual penalty alternatives as they are used by B_0.

Proof
We investigate what happens when the two paths either have or do not have a common vertex. As done in the previous proofs, we draw all vertices of the shortest path on an imaginary line from the left to the right in the same order as they are used by B_0 and enumerate them increasingly from the left to the right. As we are only interested in the vertices lying on the shortest path, we do not draw the other vertices. An edge (i, j) does not imply that the direct edge (i, j) is used, but only that i is used before j in the related alternative. Thus, the path from i to j can contain several vertices which are not part of B_0.

Case 1: Both alternatives P_1, P_2 use a common vertex i between s and t.
We color the vertex i black. Then we color all vertices on the subpaths from s to i blue and all vertices on the subpaths from i to t red. Again, Theorem 4.1.1 assures that no vertex is both red and blue.

We assume that there exists a red vertex to the left of the black one.

Analogously to the proof of Theorem 4.1.2, we find a pair (x, y) with $x < y < i$ in a way that x is blue, y is red and no other vertex between x and y is colored.

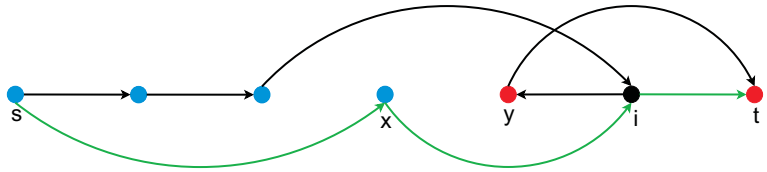


FIGURE 4.3: Case 1 of the Proof of Theorem 4.1.3, Assumption

We denote $X \to Y$ as the path from x to y as it was used in B_0 and $w(X \to Y)$ as its weight. Because of the optimality of B_0 we have:

$$w(X \to Y) < w(x \to \cdots \to i \to \cdots \to y)$$

Thus, the following pair of alternatives is better:

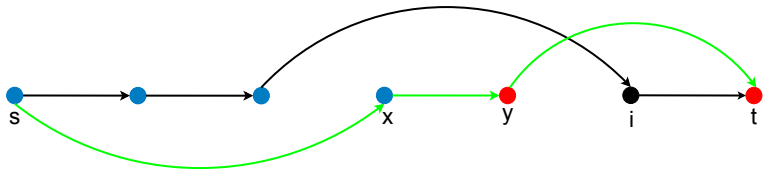


FIGURE 4.4: Case 1 of the Proof of Theorem 4.1.3, Modified Paths

This shows that the pair of alternatives P_1, P_2 cannot be optimal and thus no red vertex can exist to the left of the black one. By symmetry, there cannot exist any blue vertex to the right of the black one.

Thus, every vertex which was used before i in the optimal solution B_0 is used before it in all mutual penalty alternatives. Analogously, every vertex which was used after i in the shortest path is used after i in all alternatives.

Case 2: There does not exist any common vertex except s and t in both alternatives.

This means that both alternatives use only two common vertices: s and t. We color all edges and vertices which are used by B_1 red and all edges and vertices which are used by B_2 blue. The vertices s and t are understood as red and blue at the same time.

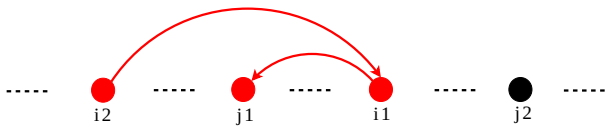
We assume that a pair (i, j) exists with the property that i is used before j by the red path but i lies to the right of j. We choose one of those pairs (i, j) with a maximum difference $i - j$ and denote these vertices i_1 and j_1. Without loss of generality there do not exist any two blue vertices a and b with $a < b$, $b - a > i_1 - j_1$ and the property that b is used before a in at least one of our alternatives. Otherwise we switch the colors of B_1 and B_2 and choose i_1 and j_1 in the recolored graph as described before.

Now, we denote the next colored vertex to the left of j_1 as i_2 and the next colored vertex to the right of i_1 as j_2. It is assured that $i_1 \neq t$ and

$j_1 \neq s$, otherwise the paths would contain a positive circuit. As s and t are also colored, i_2 and j_2 exist. The following figure shows the four important vertices. The dots mean that there can exist some more vertices in between. The red arrow means that i_1 is used before j_1 in the path. It does not mean that the direct edge (i_1, j_1) must be used.

Concerning the colors of i_2 and j_2 we study three subcases:

Subcase 2.(i) i_2 is red.
As i_1 and j_1 are the red vertices with the maximum difference $i_1 - j_1$, the vertex i_2 cannot be used after i_1 by the red $s-t$-path. Otherwise the vertices i_1 and i_2 would also fulfill our conditions and they had a larger difference than i_1 and j_1. Thus, the vertex i_2 must be used before i_1.

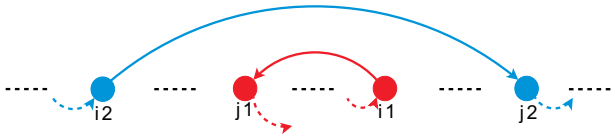


Here no edge of the subpath $i_2 \to j_1$ of B_0 is used by any alternative, otherwise there would be any vertex between i_2 and j_1 colored. As B_0 is the unique shortest path, we get a better alternative by replacing the subpath from i_2 to j_1 (using i_1) by the corresponding subpath of B_0.

Thus, our assumption was wrong.

Subcase 2.(ii) j_2 is red.
By symmetry to Subcase 2.(i), this case cannot occur, either.

Subcase 2.(iii) i_2 and j_2 are blue.
As i_2 and j_2 cannot be red, this is the only remaining case. Because of the choice of i_1 and j_1, we know that i_2 is used in the path before j_2. Every graph with the properties claimed above looks like the following one:



By cutting these alternatives in the vertices i_1, i_2, j_1 and j_2 we get the six subpaths $s \to i_1$, $i_1 \to j_1$, $j_1 \to t$, $s \to i_2$, $i_2 \to j_2$ and $j_2 \to t$.

With these subpaths and the new subpaths $i_2 \to j_1$ and $i_1 \to j_2$ we can construct the following two paths:

$s \to i_1 \to j_2 \to t$ and
$s \to i_2 \to j_1 \to t$.

This new pair of alternatives is drawn green and orange in the following figure.

The subpaths $i_1 \to j_1$ and $i_2 \to j_2$ are not used anymore. As the shortest path from i_2 to j_2 uses the subpaths $i_2 \to j_1$, $j_1 \to i_1$ and $i_1 \to j_2$, we especially know that $w(i_2 \to j_1) + w(i_1 \to j_2) < w(i_2 \to j_2)$ holds. Here, $i_2 \to j_1$ and $i_1 \to j_2$ are subpaths of B_0 and $i_2 \to j_2$ is the subpath of B_2. Thus, it especially holds:

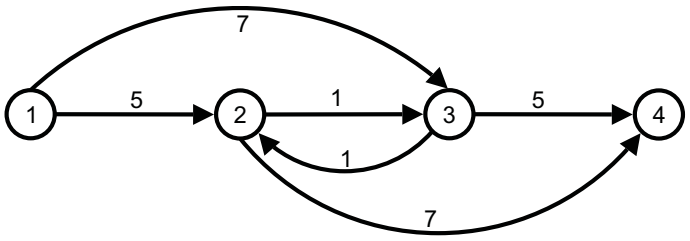
$w(i_2 \to j_1) + w(i_1 \to j_2) < w(i_2 \to j_2) + w(i_1 \to j_1)$.

That shows that our modification gives us a better pair of alternatives and thus the assumption is wrong.

The combination of these two cases proves the theorem. The first case shows that we can cut the paths in the common vertices and study the subpaths separately. The second case proves that the theorem holds for the resulting subgraphs. ■

For a pair of alternatives which is generated by the simple penalty method only Theorem 4.1.2 holds. This property holds by the definition of the simple penalty method. The statements of the other two Theorems 4.1.1 and 4.1.3 do not hold.

Example 4.1.4.
To show that the properties of the Theorems 4.1.1 and 4.1.3 do not hold for the simple penalty method, we take a look at the following graph:



The optimal solution is $1 \to 2 \to 3 \to 4$.
For $\varepsilon = 0$ the penalty alternative is also $1 \to 2 \to 3 \to 4$.
For $\varepsilon = 1$ the penalty alternative is $1 \to 3 \to 2 \to 4$.

This contradicts the statement of Theorem 4.1.1 because the vertices 2 and 3 are used in a different order by B_0 and B_ε for $\varepsilon = 1$. Because of the order of these two vertices, this example also contradicts the statement of Theorem 4.1.3 for the simple penalty method.

This shows that these two properties hold for the mutual penalty method, but not for the simple penalty method.

4.1.2 A Fast Algorithm for the Generation of Mutual Shortest Paths

In his doctoral dissertation [Sch 2003, pp. 20-27], Schwarz introduced the mutual penalty method with the help of linear programming. The biggest disadvantage of the mutual penalty method was that the generation of alternatives was a difficult and slow process. In this section we give a fast algorithm for the generation of pairs of mutual penalty alternatives. The successive shortest path algorithm from Section 2.4 will be used. To be able to use this algorithm, we have to generate a graph in which the minimum cost flow gives us the pair of mutual penalty alternatives.

As base of our minimum cost flow problem we take the graph of our shortest path problem. Then, we introduce a second edge for every edge of G. Let (i, j) be an edge of G, then this second edge gets weight $(1 + \varepsilon) \cdot w(i, j)$. All edges that were introduced so far have a capacity of one. We can interpret a second edge as the fact, that it only will be used if the original edge is also used, so we get weight $(2 + \varepsilon)\, w\,(e)$ if e is used by both paths. Now it remains to limit the maximum flow to two, because we only want to get two alternatives. Therefore we insert a new start

vertex s_{new} with one outgoing edge to s with capacity two and weight zero. Then we apply the successive shortest path algorithm to this constructed flow network and get a flow of two with minimum weight. The complexity of the successive shortest path algorithm lies in the generation of two shortest paths. With Dijkstra's shortest path algorithm, the mutual penalty alternatives can be generated within runtime $O(m \cdot log(n))$. This shows that generating mutual penalty alternatives is not more time intensive than generating simple penalty alternatives.

To demonstrate the different steps of this algorithm, we give a small example.

Example 4.1.5.
We generate the mutual shortest paths for $\varepsilon = 4$ from 1 to 4 in the following graph:

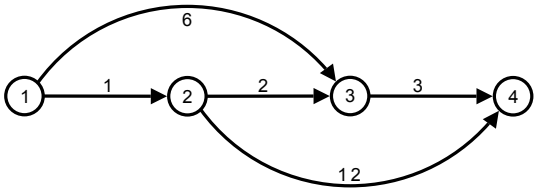


FIGURE 4.5: Example for the Generation of Mutual Shortest Paths

The first step of our algorithm is to insert a punished second edge e' for every edge e of this graph. This edge e' has a weight of $(1+\varepsilon)\, w(e)$. We also insert our new start vertex s_{new} and its outgoing edge. With this modification we get the following graph for $\varepsilon = 4$:

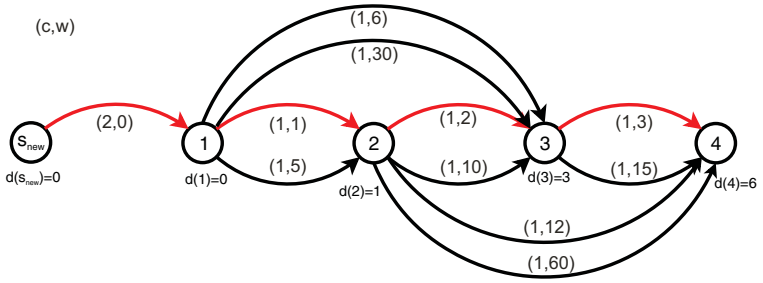


FIGURE 4.6: First Step for the Generation of Mutual Shortest Paths

The second step is the generation of the shortest paths from s_{new} to all other vertices in this modified graph. With $d(i)$ we denote the length of the shortest path from s_{new} to i. The shortest path P from s_{new} to 4 is the path $s_{new} \to 1 \to 2 \to 3 \to 4$. This path is already represented with red edges in Figure 4.6.

Now we perform the modification step. Let c^* be the smallest capacity on the shortest path from s_{new} to 4 (here $c^* = 1$). Then we set:

- $w(x,y) := w(x,y) + d(x) - d(y)$ for all $(x,y) \in E$,
- $f(x,y) := f(x,y) + c^*$ for all forward edges $(x,y) \in P$,
- $f(x,y) := f(x,y) - c^*$ for all backward edges $(y,x) \in P$,
- $c(x,y) := c(x,y) - c^*$ for all $(x,y) \in P$.
  Delete all edges (x,y) which have a capacity $c(x,y) = 0$ after this modification.
- For all edges $(x,y) \in P$ do:
  - If the backward edge (y,x) already exists, then we set $c(y,x) := c(y,x) + c^*$,
  - otherwise we introduce the backward edge (y,x) and set $c(y,x) := c^*$ and $w(y,x) = 0$.

Thus, we get $f(s_{new},1) = f(1,2) = f(2,3) = f(3,4) = 1$ and all other edges have a flow of zero. After this modification the flow network looks as follows:

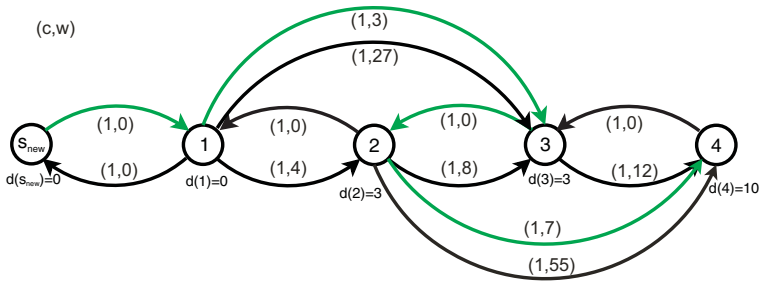


FIGURE 4.7: Second Iteration of Successive Shortest Path Algorithm

Now we generate the shortest paths in the new graph. The shortest path from s_{new} to 4 is the path $s_{new} \to 1 \to 3 \to 2 \to 4$ and is already colored green in Figure 4.7. Performing the modification step again, we get $f(s_{new},1) = 2, f(1,2) = f(1,3) = f(2,4) = f(3,4) = 1$. All other edges – especially the edge $(2,3)$ – have a flow of zero. The flow of $(2,3)$ was reduced in the second iteration, because the backward edge $(3,2)$ was used. The modified flow network after this second iteration is the following one:

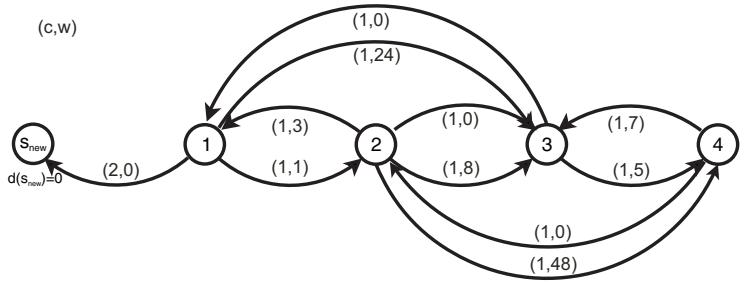


FIGURE 4.8: Flow Network after the Second Iteration of the Successive Shortest Path Algorithm

In this network there does not exist any path from s_{new} to 4, thus the algorithm terminates. Now we transform this flow network back to our basic graph by deleting the vertex s_{new} and deleting the second edges. If such a second edge is used by a pair of alternatives, then both paths use the original edge of the basic graph. In our example we get the following pair of alternatives (printed in red and blue):

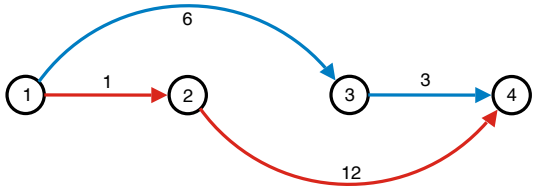


FIGURE 4.9: Mutual Shortest Paths for $\varepsilon = 4$

We can easily modify this algorithm to get k mutual penalty alternatives instead of two. The simple way would be to introduce not one punished edge, but $k-1$ ones with different punishments. But this graph can be reduced by allowing only one edge (i,j) for every pair (i,j) - the shortest one amongst all multiple edges. After an augmenting step a forward edge e that was used is replaced by a stronger punished edge. Amongst all these stronger punished edges (if they exist), we take the shortest one. Analogously, the backwards edge for e is replaced by a stronger punished edge in the same way. If a backward edge is used, we replace it and its backwards edge again (if they exist). But here we replace it by a weaker punished edge. With this modification the number of edges is independent of k. Thus, the Dijkstra algorithm has the same runtime as before. But we have to generate k shortest paths instead of two. Altogether, we can generate k-mutual penalty alternatives within runtime $O(k \cdot m \cdot log(n))$.

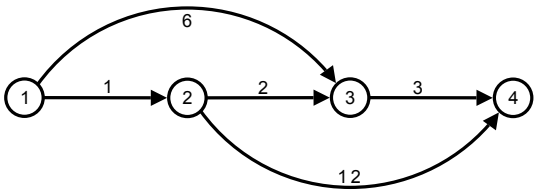
Theorem 4.1.6.
If the shortest $s-t$-path uses the edges $s \to i$ and $j \to t$ then there exists a pair of mutual penalty alternatives for all penalty parameters which uses these edges.

Proof
We prove this theorem with the help of our successive shortest path algorithm. In the first step the edges $s \to i$ and $j \to t$ get a flow of one. If these two edges are not used by a pair of mutual penalty alternatives, the backward edges $i \to s$ or $t \to j$ have to be used. But using these backward edges causes circuits $s \to \ldots \to i \to s$ or $t \to j \to \ldots \to t$. As all edges weights are nonnegative, the path without these circuits is better. ■

4.1.3 An Upper Bound for the Number of Threshold Parameters

In Chapter 3 we showed that every new simple penalty alternative contains at least one new element. This property leads to an upper bound for the number of threshold parameters. Unfortunately, this property does not hold for the mutual penalty method as the following remark shows.

Remark 4.1.7.
We take the graph which was already used to demonstrate the algorithm for the generation of mutual shortest paths:



We get the following pairs of mutual penalty alternatives:
For $0 \leqslant \varepsilon \leqslant 1$ the paths $1 \to 2 \to 3 \to 4$ and $1 \to 2 \to 3 \to 4$ are optimal.
For $1 \leqslant \varepsilon \leqslant 2$ the paths $1 \to 2 \to 3 \to 4$ and $1 \to 3 \to 4$ are optimal.
For $2 \leqslant \varepsilon \leqslant 3$ the paths $1 \to 2 \to 3 \to 4$ and $1 \to 2 \to 4$ are optimal.
And for $\varepsilon \geqslant 3$ the paths $1 \to 2 \to 4$ and $1 \to 3 \to 4$ are optimal.

The pair of alternatives which is new at $\varepsilon = 3$ does not use any new edge.

As a new pair of alternatives does not have to use a new edge we need another approach to find a good upper bound for the number of threshold parameters. Fortunately, we can give a similar property for the mutual shortest paths which allows us to formulate an upper bound. We want to motivate this with the following lemma.

Lemma 4.1.8.
A pair of mutual penalty alternatives for the shortest path problem can be identified by the edges which are used twice. By knowing only these edges, we can easily construct the complete pairs of alternatives.

Proof
Theorem 4.1.2 shows that edges can only be used twice in any alternative – and can be punished – if they are part of the shortest path. Theorem 4.1.1 gives us the property that all vertices lying on both paths are used in the same order by both paths. Theorem 4.1.3 assures that the vertices lying on the shortest $s-t$-path are used in the same order by every pair of mutual penalty alternative (especially in the same order as used by the shortest path).

Knowing the punished edges, we can construct the related pair of mutual penalty alternatives easily without knowing the penalty parameter ε. First, we combine punished subpaths to segments, i.e. having the punished edges (i,j) and (j,k) we build a segment (i,k). We can build these segments, because Theorem 4.1.3 assures that these edges are used in this order. In this way also more than two edges can be combined to a segment. Then, we sort the punished segments in the same order as they are used by the shortest path. It remains to fill the gaps between these segments.

Let $i_1, i_2, \ldots, i_n$ be the shortest $s-t$-path. Let (i_a, i_b) and (i_e, i_f) with $b < e$ be two segments which are used by our pair of mutual penalty alternatives. These segments are chosen in a way that no segment (i_c, i_d) with $b < c < d < e$ is used by our alternatives. That means, the gap from i_b to i_e has to be filled. Therefore we only have to generate two disjoint paths P_1 and P_2 from i_b to i_e with the minimum weight $w(P_1) + w(P_2)$. These paths can be generated with the successive shortest path algorithm of Section 4.1.2 by forbidding all edges which were used by the segments. In this way we can fill all gaps and get our pair of alternatives. ■

With the help of this lemma we get an analogous property for the mutual penalty method as we had for the simple penalty method. We show that every new pair of mutual shortest paths contains at least one new gap. For simplicity, we denote a gap (i,j) as unpunished subpath $i \to j$. That means that both subpaths from i to j do not use any punished second edge.

Theorem 4.1.9.
Let G be an arbitrary directed graph with n vertices and let s be the start vertex and t the target vertex. Let $w : E \to \mathbb{R}^+$ be a positive weight function on E and let $P = (s = i_1, i_2, \ldots i_{k-1}, i_k = t)$ be the unique shortest $s-t$-path. Then, at most $\frac{k^2}{2} - \frac{k}{2}$ threshold parameters are possible.

Proof
Like in the proof of Theorem 3.1.1, we reduce our graph G to a graph $\tilde{G} = (\tilde{V}, \tilde{E})$. But the reduction works different to the reduction for the simple penalty method. Here, we construct $\tilde{G}$ in a way that every new alternative uses at least one new edge in $\tilde{G}$. This shows that the difference between G and $\tilde{G}$ is quite larger than for the simple penalty method, because Remark 4.1.7 showed that this property does not hold for G. Another difference is that we construct $\tilde{G}$ in a way that a shortest $s-t$-path in $\tilde{G}$ will determine our pair of alternatives. Thus, we do not have to generate two alternatives simultaneously.

The complexity of this proof lies in the construction of the graph $\tilde{G}$. With the help of this construction an algorithm for the generation of mutual penalty alternatives could be formulated: We transform G to $\tilde{G}$ and then we apply a shortest path algorithm to $\tilde{G}$. However, generating $\tilde{G}$ is very time consuming as we have to solve $\frac{k^2}{2} - \frac{k}{2}$ minimum cost flow problems.

We start with $\tilde{V} = \{s = i_1, i_2, \dots i_{k-1}, i_k = t\}$ and insert the edges $i_j \to i_{j+1}$, $1 \leqslant j \leqslant k-1$ into $\tilde{G}$. These edges get weight $(2+\varepsilon) \cdot w(i_j, i_{j+1})$, because using these edges in a pair of alternatives means that both paths use the edge (i_j, i_{j+1}). Now, we have to care about the case that both alternatives use different edges. Therefore, we generate the shortest pair of disjoint $i_o - i_p$-paths for every pair (i_o, i_p) of vertices of P with i_o lying in the left of i_p. This can be done with the successive shortest path algorithm:

> We insert a new vertex s_{new} into G. This vertex s_{new} is our source and has an outgoing edge to i_o with capacity two and weight zero. Then, we choose i_p as sink and give all edges of G capacity one. For all pairs (i, j) for which a pair of disjoint $i-j$-paths exists, we insert an edge from i to j which has the sum of the lengths of both disjoint paths as weight. As these paths are disjoint, the punishment of these edges is zero. If there do not exist disjoint paths from i to j then we do not insert any unpunished edge from i to j. Theorem 4.1.1 assures, that no edge of these disjoint paths can be used by another pair of subpaths within the same alternative. Thus, also in combination with other subpaths these edges of $\tilde{G}$ will always be unpunished.

It remains to investigate the case that the pairs of alternatives are not equal, but also not disjoint. Fortunately, this case is already done, because a partially punished pair of paths can be separated in equal and disjoint pairs of subpaths. This means a path from i to j containing the subpath from o to p at which both paths are equal and the subpaths i to o and p to j at which they are disjoint can be expressed with the punished edges from o to p and with the shortest pair of disjoint paths from i to o and from p to j. Thus, we also have completed this case and can handle all possible cases.

We demonstrate this reduction with the same graph which was already used to demonstrate the reduction for the simple penalty method.

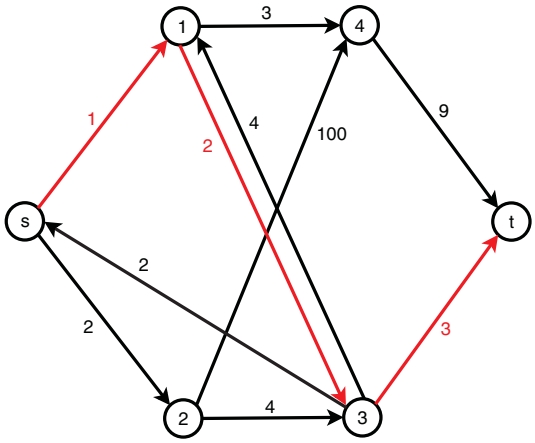
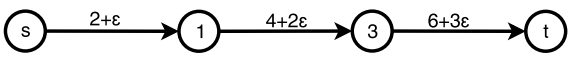


FIGURE 4.10: Construction of the Graph $\tilde{G}$ for the Mutual Penalty Method, Basic Graph

The shortest path from s to t is the path $s \to 1 \to 3 \to t$. In Figure 4.10 this path is already printed in red. Thus, the vertices s, 1, 3 and t build the vertex set $\tilde{V}$. Now, we insert the edges $s \to 1$, $1 \to 3$ and $3 \to t$ into $\tilde{G}$. They get weight $\tilde{w}(e) = (2 + \varepsilon) \cdot w(e)$, as they indicate that both subpaths use the edge e in G. After this first construction step, $\tilde{G}$ looks as follows:



Now, we insert the unpunished edges which represent disjoint subpaths:

Therefore, we generate the shortest pair of disjoint paths for every pair (i, j) of vertices of $\tilde{G}$ with the property that i is used before j by the shortest $s - t$-path in G. We start with the pair $(s, 1)$. The pair of disjoint paths from s to 1 can be determined with the successive shortest path algorithm. Here, we insert a new source s_{new} which has an outgoing edge to the start vertex of our investigated subpath (here to s) with capacity two and weight zero. All other edges have the same weights as they had in G and capacity one. Sink of this minimum cost flow problem is the target vertex of our subpath (here 1).

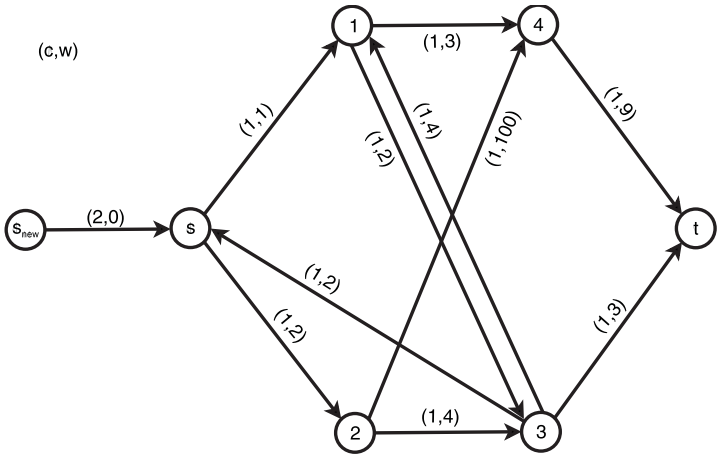


FIGURE 4.11: Construction of the Graph $\tilde{G}$ for the Mutual Penalty Method, Determination of Disjoint Subpaths from s to 1

The shortest pair of disjoint subpaths $s \to 1$ is the pair $s \to 1$ and $s \to 2 \to 3 \to 1$ with length $(1) + (2 + 2 + 4) = 11$. Thus, we insert an edge from s to 1 with weight 11 into $\tilde{G}$.

This works analogously for the subpath $1 \to t$. Here the edge from s_{new} goes to 1, the other edge capacities and weights remain unmodified. In this flow network the sink is t instead of 3. Applying the successive shortest path algorithm, we get $1 \to 3 \to t$ and $1 \to 4 \to t$ as shortest disjoint paths from 1 to t. These paths have a weight sum of $(2 + 3) + (3 + 9) = 17$. Analogously, we generate the shortest pair of disjoint paths for all other pairs of vertices. The only pair for which no pair of disjoint paths exists is the pair $(1, 3)$. Thus, $\tilde{G}$ does not contain any unpunished edge going from 1 to 3. At the end of this construction we get the following graph $\tilde{G}$.

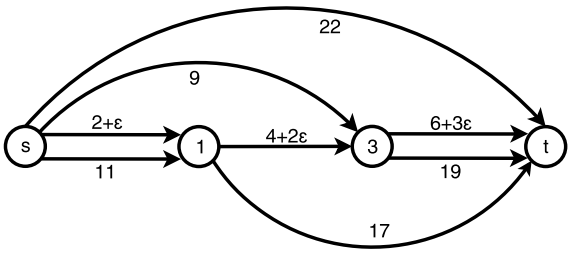


FIGURE 4.12: Construction of the Graph $\tilde{G}$ for the Mutual Penalty Method, Completed Construction

In this graph, we generate the shortest $s - t$-paths for every penalty parameter $\varepsilon \geqslant 0$. A "p" over an arrow means that the punished edge is used, otherwise the unpunished edge is used:

For $0 \leqslant \varepsilon \leqslant 1$ the path $s \xrightarrow{p} 1 \xrightarrow{p} 3 \xrightarrow{p} t$ is optimal.
For $1 \leqslant \varepsilon \leqslant 2$ the path $s \to 3 \xrightarrow{p} t$ is optimal.
For $2 \leqslant \varepsilon \leqslant 3$ the path $s \xrightarrow{p} 1 \to t$ is optimal.
And for $\varepsilon \geqslant 3$ the path $s \to t$ is optimal.

Translating these paths back to pairs of alternatives in G, we get:

For $0 \leqslant \varepsilon \leqslant 1$ the paths $s \to 1 \to 3 \to t$ and $s \to 1 \to 3 \to t$ are optimal.
For $1 \leqslant \varepsilon \leqslant 2$ the paths $s \to 1 \to 3 \to t$ and $s \to 2 \to 3 \to t$ are optimal.
For $2 \leqslant \varepsilon \leqslant 3$ the paths $s \to 1 \to 3 \to t$ and $s \to 1 \to 4 \to t$ are optimal.
And for $\varepsilon \geqslant 3$ the paths $s \to 1 \to 4 \to t$ and $s \to 2 \to 3 \to t$ are optimal.

After demonstrating the construction of $\tilde{G}$ we return to the proof of Theorem 4.1.9. We could use a include-exclude-algorithm like in [Doe 2008] to find an upper bound for the maximum number of penalty parameters, because the general conditions are very similar to those for simple penalty alternatives on DAGs. But for sake of simplicity we use an approach which is similar to the proof of Theorem 3.1.1. Here it is a little bit easier as we do not have edges from the right to the left. Generating the shortest $s-t$-path in $\tilde{G}$ for a given penalty parameter ε is equivalent to the generation of the mutual shortest paths for ε in G. But in $\tilde{G}$ we have an easy characterization of punished and unpunished edges and can use an intuitive shortest path algorithm like Dijkstra's algorithm instead of using the successive shortest path algorithm.

We denote P_{new} as an alternative that is new at ε^* and P_{old} as an alternative that is new at $\varepsilon < \varepsilon^*$ but still optimal at ε^*. As demonstrated in Remark 2.2.18, a threshold parameter ε^* can only appear at a penalty parameter at which the new alternative P_{new} has a smaller punished weight than the old alternative P_{old}. Now we show that this property in combination with the monotonicity properties of Theorem 2.2.3 and the convexity properties of Theorem 2.2.4 cause that a new alternative in $\tilde{G}$ contains at least one new edge. Therefore, we investigate the following two cases:

(*i*) The direct edge (s,t) is used ($i_2, i_3, \ldots, i_{k-2}, i_{k-1}$ do not lie on P_{new}).

Then this edge is unpunished, as only edges $i_j \to i_{j+1}$, $1 \leqslant j \leqslant k-1$ are punished and the punished length of P_{new} is smaller than the punished length of all old alternatives. The convexity property of Theorem 2.2.9 assures, that this edge is new. If it was old, the alternative would also be optimal for a smaller penalty parameter and also for all penalty parameters in between, due to convexity. Thus, ε^* would not be a threshold parameter. In case of $\tilde{V} = \{s,t\}$, we also use the unpunished edge, because the punishment decreases.

(*ii*) The alternative P_{new} uses the interior vertices $i_{j_1}, i_{j_2}, \ldots, i_{j_p}$ in this order.

As the punished weight of P_{new} is smaller than the punished weight of all old alternatives, the punished weight of at least one of its subpaths is smaller than the punished weight of the corresponding subpath in all old alternatives. The monotonicity theorem (Theorem 2.2.8) assures that the punishment of these subpaths decreases weakly for an increasing penalty parameter and Remark 2.2.18 assures that the punishment of at least one subpath decreases strictly at a threshold parameter. We choose one of the subpaths with decreasing punishment (the punished length of it is smaller in P_{new} than its punished length in every old alternative). Let x be the start vertex of this subpath and let y be its target vertex. If the direct edge $x \to y$ is used, this edge is new (case (*i*)) and the statement holds. Otherwise we denote $\mathcal{P}_{old}$ as the set of alternatives that are new at $\varepsilon < \varepsilon^*$ with the following properties:

- these alternatives contain a subpath from x to y and
- their subpaths from x to y use common interior vertices with the subpath from x to y in P_{new}

We redefine P_{old} as newest alternative amongst all alternatives of $\mathcal{P}_{old}$.

Let $x = i_{l_1}, i_{l_2}, \ldots, i_{l_q} = y$ be the vertices that are used by P_{new} and P_{old} in this order. As the punishment of P_{new} decreases at ε^*, it also decreases in at least one of the subpaths $i_{l_x} \to i_{l_{x+1}}$, $x < q$. We investigate one of these subpaths $i_{l_x} \to i_{l_{x+1}}$ with decreasing punishment recursively. After a few recursions we will come to case (*i*) and get a pair of vertices (i, j), $i < j$ with the property that the path from i to j was punished so far and is unpunished now. The convexity properties of Theorem 2.2.9 assure that this edge is new. If an alternative used this unpunished edge before and P_{new} uses it, then P_{old} also had to use it. But this contradicts our choice of (i, j). Thus, the direct edge (i, j) is used by P_{new} and it is new at ε^*.

As every new alternative contains a new edge (i, j) with $i < j$, we directly get an upper bound for the number of threshold parameters. The vertex s has at most $k - 1$ unpunished outgoing edges, i_2 has at most $k - 2$ unpunished outgoing edges and so on. As there exist at most $\sum_{i=1}^{k-1} i = \frac{k^2}{2} - \frac{k}{2}$ edges which can be new for a penalty parameter and as every new alternative in $\tilde{G}$ contains at least on new edge, we cannot get more than $\frac{k^2}{2} - \frac{k}{2}$ threshold parameters. ■

As for the simple penalty method, this bound also holds for multigraphs. The generation of $\tilde{G}$ might be more time consuming, but the principle is the same. The successive shortest path algorithm does not have any troubles with multiple edges or loops. Thus, at most $\frac{n^2}{2} - \frac{n}{2}$ threshold parameters are possible for mutual shortest paths on multigraphs.

Remark 4.1.10.
In the previous theorem we assumed that the shortest $s-t$-path is unique. Now we show that this claim only has a technical background and also problem instances with two or more shortest paths fit our upper bound for the number of threshold parameters. Let $P_1, P_2, \ldots, P_k$ be the shortest $s-t$-paths in G. Then, we choose one of them as the shortest path we want to use in the theorem above, for example P_1. Now, we modify the edge lengths minimally to get a graph with a unique shortest path. Therefore, we simply add extra weights to all edges which are in $P_2, \ldots, P_k$ but not in P_1. That means, we set $\hat{w}(e) = w(e) + \xi$ for all $e \in \left(\bigcup_{i=2}^{k} P_i\right) \setminus P_1$ and for some $\xi > 0$.

With this modification we get a graph $\hat{G}$ with the unique shortest path P_1. If the generated alternative for the problem with modified weights is worse than the generated alternative in the original problem for any penalty parameter ε, then we denote their weight difference in G with Δ. Choosing $\xi < \frac{\Delta}{n \cdot (1+\varepsilon)}$ solves this problem. So we can choose our $\xi > 0$ in a way that the set of alternatives in the modified problem contains an optimal alternative for every penalty parameter ε in the original problem.

4.1.4 A Lower Bound for the Maximum Number of Threshold Parameters

Example 4.1.11.
To show that the maximum number of threshold parameters lies in the class $\Theta(n^2)$, we construct an example with $\frac{n^2}{4} - 1$ threshold parameters for the case $n = 6$ and then we generalize it for an arbitrary n.

The idea of this example is that $B_{1(\varepsilon)}$ is the path $1 \to 2 \to 3 \to \cdots \to n-1 \to n$ which uses all vertices. The sequence of the other paths $B_{2(\varepsilon)}$ for increasing ε is the same sequence as it is given in Example A.1.1 in the appendix. Thus, the sequence of the alternatives $B_{2(\varepsilon)}$ for $n = 6$ is the following one:

$\varepsilon \in$	path
$[0, 9]$	$1 \to 2 \to 3 \to 4 \to 5 \to 6$
$[9, 10]$	$1 \to 2 \to 3 \longrightarrow 5 \to 6$
$[10, 11]$	$1 \to 2 \to 3 \longrightarrow 6$
$[11, 12]$	$1 \to 2 \longrightarrow 4 \to 5 \to 6$
$[12, 13]$	$1 \to 2 \longrightarrow 5 \to 6$
$[13, 14]$	$1 \to 2 \longrightarrow 6$
$[14, 15]$	$1 \longrightarrow 4 \to 5 \to 6$
$[15, 16]$	$1 \longrightarrow 5 \to 6$
$[16, \infty]$	$1 \longrightarrow 6$

TABLE 4.1: All 9 Possible Paths for the Mutual Shortest Path Problem, with $1 \to 2 \to 3 \to 4 \to 5 \to 6$ as Second Alternative

For this construction we make the following two conventions (for an arbitrary n):

1. $w(i, i+1) = \begin{cases} 1 & \text{for } i > \left\lfloor \frac{n}{2} \right\rfloor \\ \left\lceil \frac{n}{2} \right\rceil & \text{for } i \leqslant \left\lfloor \frac{n}{2} \right\rfloor \end{cases}$

2. The first threshold parameter is $\varepsilon_1 = \left\lfloor \frac{n^2}{4} \right\rfloor$. For the other threshold parameters we choose $\varepsilon_{i+1} = \varepsilon_i + 1$ for $i > 1$. So, we ensure that $B_{1(\varepsilon)}$ is $1 \to 2 \to 3 \to \cdots \to n-1 \to n$ for every penalty parameter ε.

The first default assures that a new pair of alternatives for every threshold parameter $\varepsilon > \left\lfloor \frac{n^2}{4} \right\rfloor$ has a punished weight, which is exactly one weight unit smaller than the punished weight of the pair of alternatives which is optimal for $\tilde{\varepsilon} \in [\varepsilon - 1, \varepsilon]$. The second default will be explained right after the construction of the example for an arbitrary number of vertices.

As described above we choose $w(1,2) = w(2,3) = w(3,4) = 3$ and $w(4,5) = w(5,6) = 1$. All other weights are calculated analogously to the examples before. For example for $\varepsilon = 9$ we have:

$$\begin{aligned} (1+\varepsilon)\left[w(1,2) + w(2,3) + w(3,4) + w(4,5) + w(5,6)\right] &= (1+\varepsilon)\left[w(1,2) + w(2,3) + w(5,6)\right] + w(3,5) \\ (1+9)\left[w(3,4) + w(4,5)\right] &= w(3,5) \\ 10 \cdot (3+1) &= w(3,5). \end{aligned}$$

Thus, we get $w(3,5) = 40$. Calculating all other weights, the following graph comes up:

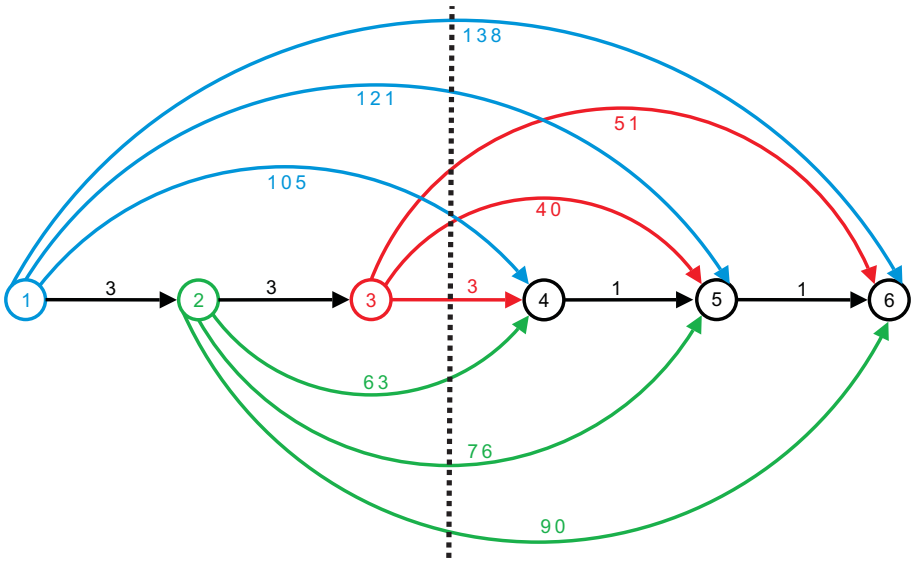


FIGURE 4.13: A Graph with 8 Threshold Parameters for the Mutual Shortest Path Problem

Generalization of the Example for an Arbitrary n

So far, we only have an example for $n = 6$ with many threshold parameters. But this example can be generalized to graphs with n vertices for an arbitrary n. Let the vertices be denoted with the numbers 1 up to n from the left to the right and let 1 be the starting vertex and n be the target vertex. We get $\frac{n^2}{4} - 1$ threshold parameters by assigning the following weights to the edges:

$$w(i,j) = \begin{cases} 1 & \text{if } j = i+1 \text{ and } \left\lfloor \frac{n}{2} \right\rfloor < i < n, \\ \left\lceil \frac{n}{2} \right\rceil & \text{if } j = i+1 \text{ and } j \leqslant \left\lfloor \frac{n}{2} \right\rfloor, \\ \frac{x_{i,j}(n) \cdot (x_{i,j}(n)-1)}{2} + \left(\left\lceil \frac{n}{2} \right\rceil + x_{i,j}(n) \right) \cdot \left(\left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) & \text{if } i \leqslant \left\lfloor \frac{n}{2} \right\rfloor,\ j > \left\lfloor \frac{n}{2} \right\rfloor \text{ and } j \geqslant i+2, \\ \infty & \text{otherwise,} \end{cases}$$

with $x_{i,j}(n) := \left(\left\lfloor \frac{n}{2} \right\rfloor - i \right) \left\lceil \frac{n}{2} \right\rceil + j - \left\lfloor \frac{n}{2} \right\rfloor - 1$.

The Background of the Second Convention

Every path can be identified by the edge (i,j) with $i \leqslant \left\lfloor \frac{n}{2} \right\rfloor, j > \left\lfloor \frac{n}{2} \right\rfloor$ that is used. Now we show that one of the alternatives uses the edge $\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor + 1 \right)$ if we choose $\varepsilon_1 = \left\lfloor \frac{n^2}{4} \right\rfloor$ as smallest threshold parameter.

- Choosing $\varepsilon_1 = \left\lfloor \frac{n^2}{4} \right\rfloor$ and $\varepsilon_{i+1} = \varepsilon_i + 1$ for $i > 1$ gives us $\varepsilon = \left\lfloor \frac{n^2}{2} \right\rfloor$ as largest threshold parameter.
- If the pair of alternatives uses the edge $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ then we can separate $B_{1(\varepsilon)} \cup B_{2(\varepsilon)}$ into $B_3 \cup B_4$ in a way that B_3 is the path $1 \to 2 \to 3 \to \cdots \to n-1 \to n$. This directly would give us the wanted result.
- It is assured that at least one alternative uses the vertex $\left\lfloor \frac{n}{2} \right\rfloor$.
  Let the alternatives use the edges (i,j) and (k,l) with $i \leqslant k < \left\lfloor \frac{n}{2} \right\rfloor$ and $j, l > \left\lfloor \frac{n}{2} \right\rfloor$. Then, the weight of the pair of alternatives can be improved by replacing the path $k \to l$ by $k \to (k+1) \to (k+2) \to \ldots \to \left\lfloor \frac{n}{2} \right\rfloor \to l$.
- The correctness of the first $\left\lfloor \frac{n}{2} \right\rfloor$ pairs of alternatives is easy to verify. All other pairs of alternatives have a larger unpunished weight than the paths $1 \to 2 \to \ldots \to (n-1) \to n$ and $1 \to 2 \to \ldots \to \left\lfloor \frac{n}{2} \right\rfloor \to n$, which is optimal for $\varepsilon = \left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$. Thus, these other pairs of paths cannot be optimal for $\varepsilon \leqslant \left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$.
- For all larger penalty parameters $\varepsilon \geqslant \left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$ the edge $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ or the edge $\left(\left\lfloor \frac{n}{2} \right\rfloor, n\right)$ is used.

  For $\varepsilon \geqslant \left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$ the edge $\left(\left\lfloor \frac{n}{2} \right\rfloor, n\right)$ is shorter than all subpaths using an unpunished edge from $\left\lfloor \frac{n}{2} \right\rfloor$ to a vertex $i > \left\lfloor \frac{n}{2} \right\rfloor + 1$ and the punished edges $(i, i+1), (i+1, i+2), \ldots (n-1, n)$ (because of Theorem 2.2.8).

  If an edge from $\left\lfloor \frac{n}{2} \right\rfloor$ to i with $\left\lfloor \frac{n}{2} \right\rfloor + 1 < i < n$ is used for $\varepsilon \geqslant \left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$, then the successive shortest path algorithm uses either the edges $(i, i+1), (i+1, i+2), \ldots, (n-1, n)$ or the edges $(i, i-1), \ldots, (\left\lfloor \frac{n}{2} \right\rfloor + 1, \left\lfloor \frac{n}{2} \right\rfloor)$ in the path from i to n. In the first case the direct edge $(\left\lfloor \frac{n}{2} \right\rfloor, n)$ is better as described above. In the second case we have a nonnegative circuit $\left\lfloor \frac{n}{2} \right\rfloor \to i \to \cdots \to \left\lfloor \frac{n}{2} \right\rfloor$.

  Thus, we do not use any edge $(\left\lfloor \frac{n}{2} \right\rfloor, i)$ with $\left\lfloor \frac{n}{2} \right\rfloor + 1 < i < n$.

So far, we know that one of the edges $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ and $\left(\left\lfloor \frac{n}{2} \right\rfloor, n\right)$ is used by every pair of alternatives. It remains to show that every pair of alternatives uses $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$.

For the following investigations we use a simplified successive shortest path algorithm to generate our pairs of alternatives. It works as follows:

0. Introduce the punished second edges
1. Generate the shortest $s-t$-path P.
2. Multiply the weights of all edges of P by -1 and switch their direction.
3. Generate the shortest $s-t$-path in the modified graph.

The edge set of the pair of alternatives is the set of all edges which are used in exactly one direction. This algorithm works analogously to our successive shortest path algorithm. As we do not modify the edge weights, we directly use the weights of the example for an arbitrary n.

The shortest path P in G is $1 \to 2 \to 3 \to \ldots \to (n-1) \to n$. As this path is found in the first step of the algorithm, we switch the direction of all edges $(i, i+1)$ and multiply their weight by -1.

If the pair of alternatives uses the edge $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$, then the backward edge $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1, \left\lfloor \frac{n}{2} \right\rfloor\right)$ will not be used in the third step of our algorithm. Suffice it to show that it does not make sense to use this backward edge for any $\varepsilon \in \left[\left\lfloor \frac{n^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n^2}{2} \right\rfloor\right]$. For $\varepsilon = \frac{n^2}{2}$ we show that the path from $\left\lfloor \frac{n}{2} \right\rfloor + 1$ to n does not use the backward edge $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1, \left\lfloor \frac{n}{2} \right\rfloor\right)$. Therefore, we show that the path $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left\lfloor \frac{n}{2} \right\rfloor \to n$ is longer than the path $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 2\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 3\right) \to \ldots \to n$ for $\varepsilon = \frac{n^2}{2}$.

Using the weights of the example for arbitrary n, the path $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left\lfloor \frac{n}{2} \right\rfloor \to n$ has length

$$\begin{aligned} -\left(\left\lceil \frac{n}{2} \right\rceil\right) &+ \frac{\left(n - \left\lfloor \frac{n}{2} \right\rfloor - 1\right)\left(n - \left\lfloor \frac{n}{2} \right\rfloor - 2\right)}{2} + \left(\left\lceil \frac{n}{2} \right\rceil + n - \left\lfloor \frac{n}{2} \right\rfloor - 1\right)\left(\left\lfloor \frac{n^2}{4} \right\rfloor + 1\right) \\ &\geqslant -\left(\frac{n}{2} + 1\right) + \frac{\left(n - \frac{n}{2} - 1\right)\left(n - \frac{n}{2} - 2\right)}{2} + \left(\frac{n}{2} + n - \frac{n}{2} - 1\right)\left(\frac{n^2}{4}\right) \\ &= \frac{n^3}{4} - \frac{n^2}{8} - \frac{5n}{4}. \end{aligned}$$

The path $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 2\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 3\right) \to \ldots \to n$ uses $\left(n - \left\lfloor \frac{n}{2} \right\rfloor - 2\right)$ edges of weight $(1 + \varepsilon) = \left(1 + \frac{n^2}{2}\right)$ (for $\varepsilon = \frac{n^2}{2}$). Thus, its length is

$$\left(n - \left\lfloor \frac{n}{2} \right\rfloor - 2\right)\left(1 + \frac{n^2}{2}\right) \leqslant \left(\frac{n}{2} - 1\right)\left(1 + \frac{n^2}{2}\right) \leqslant \frac{n^3}{4} - \frac{n^2}{2} + \frac{n}{2}.$$

The weight difference of both paths is $(\frac{n^3}{4} - \frac{n^2}{8} - \frac{5n}{4}) - (\frac{n^3}{4} - \frac{n^2}{2} + \frac{n}{2}) = \frac{3n^2}{8} - \frac{7n}{4}$. As this difference is larger than zero for $n > 4$, it does not make sense to use the backward edge $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1, \left\lfloor \frac{n}{2} \right\rfloor\right)$ for $\varepsilon = \left\lfloor \frac{n^2}{2} \right\rfloor$ and $n > 4$. In case of $\varepsilon < \frac{n^2}{2}$ this holds more than ever, because the length of $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left\lfloor \frac{n}{2} \right\rfloor \to n$ does not depend on ε and the length of $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 2\right) \to \left(\left\lfloor \frac{n}{2} \right\rfloor + 3\right) \to \ldots \to n$ decreases in ε. Because of that, all pairs of alternatives for a penalty parameter $\varepsilon \leqslant \frac{n^2}{2}$ contain the path $1 \to 2 \to 3 \to \ldots \to (n-1) \to n$. For the case $n \leqslant 4$ our construction works as well, of course. As the alternatives for $\varepsilon = \frac{n^2}{2}$ are disjoint, there cannot exist any threshold parameter for $\varepsilon > \frac{n^2}{2}$.

The combination of Example 4.1.11 and Theorem 4.1.9 gives us the lower bound $\frac{n^2}{4}-1$ and the upper bound $\frac{n^2}{2}-\frac{n}{2}$ for the maximum number of threshold parameters. Thus, this maximum number lies in the class $\Theta(n^2)$.

4.1.5 Mutual Shortest Paths on Grid Graphs

Schwarz [Sch 2003] and Sameith [Sam 2005] used grid graphs to demonstrate advantages of the penalty methods. Here, we use them again to formulate two structural properties of the pairs of the mutual penalty alternatives.

Definition 4.1.12 (Weighted Directed Grid Graph).
A weighted directed grid graph of size $n \times n$ is a graph $G = (V, E)$ with $V = \{v_{i,j} : 1 \leqslant i, j \leqslant n\}$ and $E = \{(v_{i,j}, v_{k,l}) : i, j, k, l \in \{1, \ldots, n\}$ with $0 \leqslant (k-i) \leqslant 1$, $0 \leqslant (l-j) \leqslant 1$ and $(k-i)+(l-j) = 1\}$ and a positive weight function $w : E \to \mathbb{R}^+$. Start vertex s is the lower left vertex $v_{1,1}$ and target vertex t is the upper right vertex $v_{n,n}$. In contrast to all other graphs before, the number of vertices is not n, but n^2.

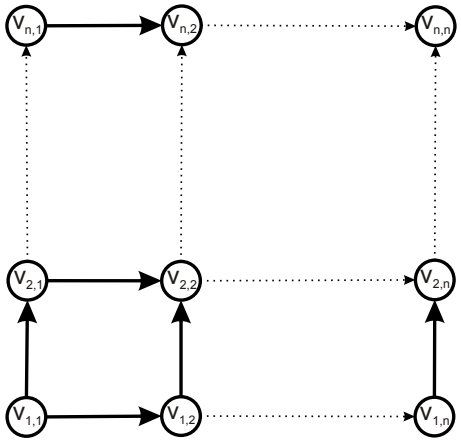


FIGURE 4.14: Directed $n \times n$ Grid Graph

The mutual penalty alternatives on grid graphs typically look like the following ones:

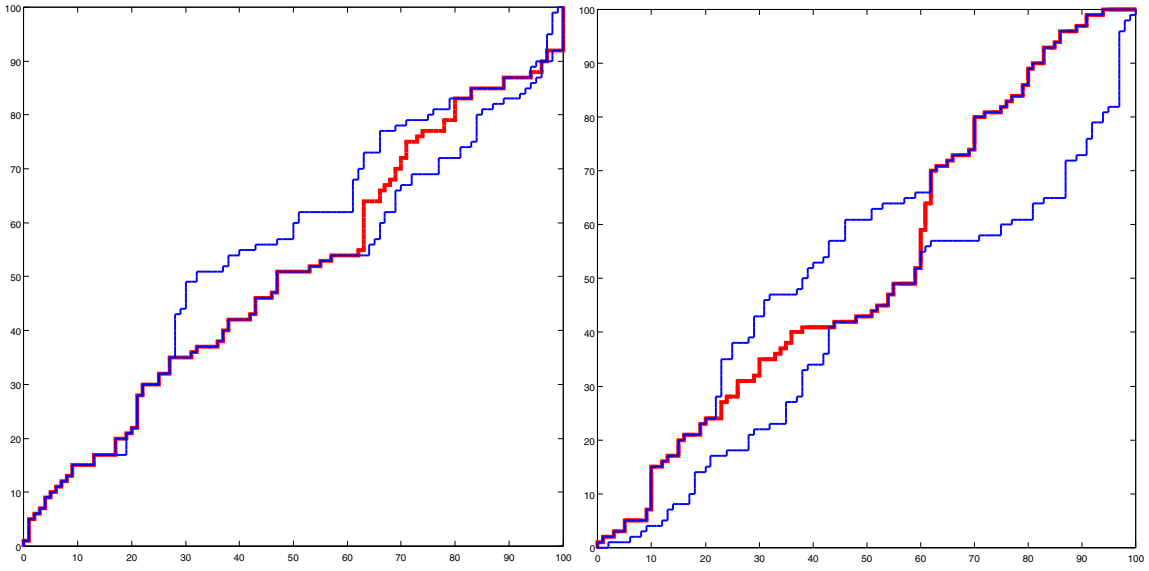


FIGURE 4.15: Pair of Mutual Penalty Alternatives (blue) (Left Figure: $\varepsilon = 0.1$, Right Figure $\varepsilon = 1$) and the Shortest Path (red) in a 100×100 Grid Graph

Figure 4.15 demonstrates two properties of the mutual penalty alternatives:

Theorem 4.1.13.
If the shortest path on directed grid graphs is unique, then:

(*i*) *The shortest path passes the grid between our two alternatives.*

(*ii*) *If the shortest path leaves the upper path at any point, then it moves towards the lower path and not directly back to the upper path. The symmetric case holds analogously.*

Proof
Let P_1 be the upper path, this means that it never moves below path P_2 and let P_2 be the lower path.

(*i*) Let the shortest path P pass the grid anywhere above path P_1. Then, there exists some vertex x where P leaves P_1 and another vertex y where P returns to P_1. The extreme case is $x = s$ and $y = t$. As P_1 is the upper path, the shortest path does not use any edge of P_1 or P_2 between the vertices x and y. Thus, we can redirect path P_1 in a way that it uses the part between x and y of the shortest path. Here, it is assured that no edge of this subpath is punished after this modification. It is also assured that the modified path P_1 is shorter after this modification since the shortest path is unique. The path P_2 does not become longer by this modification. This contradicts our assumption because the pair of alternatives P_1 and P_2 cannot be optimal. By symmetry the shortest path also cannot pass the area below P_2.

(*ii*) Now we assume that P leaves P_1 in x, returns to P_1 in y and does not meet P_2 in between. Then, we can redirect P_1 again using the subpath from x to y as used by P. As the shortest path is unique, the path P_1 becomes shorter by this modification and P_2 remains unmodified. This contradicts the optimality of P_1 and P_2. By symmetry it can also not occur that P leaves P_2 and returns to P_2 without meeting P_1 in between. ■

4.2 Valuated Matroids

Surprisingly, the mutual penalty alternatives for the shortest path problem revealed really nice properties. But the structure of valuated matroids behaves even better. As for the shortest path problem, all common elements of both alternatives are also part of the optimal base. In contrast to the shortest path method, it even holds that every element of the optimal base is part of at least one of the mutual penalty alternatives. Analogously to the simple penalty method and to [AW 1999a], we prove a structural monotonicity. With these properties we get the result that at most $rank(M)$ threshold parameters are possible. Then, show that this bound is tight by giving an example for the MST problem which exactly meets this upper bound.

Remark 4.2.1.
We have the valuation $v : E \to \mathbb{R}^- \cup \{-\infty\}$ and want to maximize the weight of the pair of alternatives. The mutual penalty method concerning valuated matroids can be understood as follows:

- We want to find two bases B_1 and B_2.
- Every element e exists twice: on the one hand as element e with weight $v(e)$ and on the other hand as punished element e' with weight $v(e') = (1+\varepsilon) \cdot v(e)$. If a base B contains e, then we can exchange e by e' and get a base again.
- An alternative cannot contain as well e as e' for any $e \in E$.
- An element e cannot be part of as well B_1 as B_2. Here, one of the bases has to use e' instead of e. Thus, we get two disjoint bases.

Theorem 4.2.2.
Let M_v be a valuated matroid with an injective weight function v. Let B be the optimal base of the matroid and let B_1 and B_2 be the mutual penalty alternatives for an arbitrary penalty parameter $\varepsilon \geqslant 0$. Then $B \subseteq B_1 \cup B_2$ holds.

Proof
For $\varepsilon = 0$ we have $B_1 = B_2 = B$ and thus, the statement holds. Now, we choose $\varepsilon > 0$. We assume that some $e^* \in B$ exists which is not part of B_1 or B_2.

The pair of alternatives (B_1, B_2) cannot be optimal if $e' \in B_1 \cup B_2$ but $e \notin B_1 \cup B_2$. Here we get a better pair of alternatives by replacing e' by e. This fact especially shows that $B_1 \cup B_2$ cannot contain more punished elements than unpunished ones. If $B_1 \cup B_2$ contains as many punished elements as unpunished ones, then $B_1 = B_2 = B$ holds.

Thus, there are more unpunished elements as punished ones in $B_1 \cup B_2$. Now we separate $B_1 \cup B_2$ into bases B_3 and B_4 in a way that B_3 does not contain any punished element. This works, because for every used punished element e' the unpunished one is used by the other alternative as shown above. That means B_4 contains all punished elements which were used by $B_1 \cup B_2$.

Let e^* be an arbitrary element in $B \setminus (B_1 \cup B_2)$. Because of $B_3 \subseteq B_1 \cup B_2$, we have $e^* \in B \setminus B_3$. Then Axiom $(V1')$ shows that there exists some $f \in B_3 \setminus B$ with:

$$v(B) + v(B_3) = v((B \setminus \{e^*\}) \cup \{f\}) + v((B_3 \setminus \{f\}) \cup \{e^*\}).$$

As B is the optimal base of M_v, we also know that $v(B) > v((B \setminus \{e^*\}) \cup \{f\})$ holds. Hence, $v(B_3) < v((B_3 \setminus \{f\}) \cup \{e^*\})$ holds, too. This shows that the exchange improves the weight of B_3 and does not affect B_4 in any way. Thus, $B_3 \cup B_4 = B_1 \cup B_2$ cannot be optimal and our assumption was wrong. ■

Theorem 4.2.3.
Let M_v be a valuated matroid with an injective weight function v. Let B be the optimal base of the matroid and let B' be the punished equivalent of B. Then, no punished element e' with $e' \notin B'$ is used by any pair of mutual penalty alternatives.

Proof
Let $B_1 \cup B_2$ be the optimal pair of alternatives for an arbitrary penalty parameter ε. Then, we separate $B_1 \cup B_2$ into bases B_3 and B_4 in a way that B_3 does not use some punished element.

We assume that any punished element $f' \in B_4 \setminus B'$ exists. Then, Axiom $(V1')$ shows that some $e' \in B' \setminus B_4$ exists with:

$$v(B') + v(B_4) = v((B' \setminus \{e'\}) \cup \{f'\}) + v((B_4 \setminus \{f'\}) \cup \{e'\}).$$

As B is the optimal base, the following inequalities hold:

$$\begin{aligned} & v(B) && > v((B \setminus \{e\}) \cup \{f\}) \\ \Leftrightarrow\ & (1+\varepsilon) \cdot v(B) && > (1+\varepsilon) \cdot v((B \setminus \{e\}) \cup \{f\}) \\ \Leftrightarrow\ & v(B') && > v((B' \setminus \{e'\}) \cup \{f'\}). \end{aligned}$$

Because of $v(B') > v((B' \setminus \{e'\}) \cup \{f'\})$ we get $v(B_4) < v((B_4 \setminus \{f'\}) \cup \{e'\})$. Otherwise we would have a contradiction to Axiom $(V1')$.

This means that we can improve the weight of B_4 by exchanging f' by e'. The base B_3 will not be affected by this exchange. So $B_3 \cup B_4 = B_1 \cup B_2$ cannot be optimal. ■

Remark 4.2.4.
Here we are not only looking for one base but two disjoint bases of a matroid. This problem was already investigated by Roskind and Tarjan [RT 1985] and by Edmonds [Edm 1965]. They defined the union of two disjoint bases as a base of a new matroid. Thus, we can also use our matroid Axioms $(V0)$ and $(V1')$ for the union of our two mutual penalty alternatives. In these publications and in Chapter 13 of [Rec 1989] fast algorithms for the generation of these disjoint bases can be found, which can be directly used to generate our pairs of mutual penalty alternatives. In these references, augmenting sequences are used to find k (here $k = 2$) disjoint bases with a maximum weight. With the help of this method we find pairs of mutual penalty alternatives.

We also get a similar structural monotonicity for the mutual penalty method like we already had for the simple penalty method. With the help of Remark 4.2.4, the proof works analogously to the proof of Theorem 3.2.1.

Theorem 4.2.5 (Structural Monotonicity).
Let $M_v = (E, v)$ be a valuated matroid with an injective weight function $v : E \to \mathbb{R}^-$ and let B_0 be the optimal base. Then the following two properties hold:

(i) For all $\delta < \varepsilon$ and all $a \in B_0 \setminus \left(B_{1(\delta)} \cup B_{2(\delta)}\right)$ it holds $a \notin \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$.

(ii) For all $\delta < \varepsilon$ and all $b \in \left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus B_0$ it holds $b \in \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$.

Proof
For the penalty parameter ε we denote the punished weight of a pair of alternatives (B_1, B_2) as $v_\varepsilon(B_1 \cup B_2) = v(B_1) + v(B_2) + \varepsilon \cdot v(B_1 \cap B_2)$. Let ε and δ be nonnegative penalty parameters with $\delta < \varepsilon$. If $B_{1(\varepsilon)} \cup B_{2(\varepsilon)}$ and $B_{1(\delta)} \cup B_{2(\delta)}$ are equal then the properties (i) and (ii) are fulfilled. Thus, we assume $B_{1(\varepsilon)} \cup B_{2(\varepsilon)} \neq B_{1(\delta)} \cup B_{2(\delta)}$. Now, we choose an arbitrary element $e \in \left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$. Due to Axiom $(V1')$ for valuated matroids of sum type, some element $f \in \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \setminus \left(B_{1(\delta)} \cup B_{2(\delta)}\right)$ exists with

$$v_\delta\left(B_{1(\delta)} \cup B_{2(\delta)}\right) + v_\delta\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \tag{4.2.1}$$
$$= v_\delta\left(\left(\left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \{e\}\right) \cup \{f\}\right) + v_\delta\left(\left(\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \setminus \{f\}\right) \cup \{e\}\right)$$

and

$$v_\varepsilon\left(B_{1(\delta)} \cup B_{2(\delta)}\right) + v_\varepsilon\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \tag{4.2.2}$$
$$= v_\varepsilon\left(\left(\left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \{e\}\right) \cup \{f\}\right) + v_\varepsilon\left(\left(\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \setminus \{f\}\right) \cup \{e\}\right).$$

As the pair of alternatives $B_{1(\delta)} \cup B_{2(\delta)}$ is optimal for the penalty parameter δ and the pair $B_{1(\varepsilon)} \cup B_{2(\varepsilon)}$ is optimal for ε, we have

$$\begin{aligned} v_\delta\left(B_{1(\delta)} \cup B_{2(\delta)}\right) &\geqslant v_\delta\left(\left(\left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \{e\}\right) \cup \{f\}\right) &&\Rightarrow v_\delta(e) \geqslant v_\delta(f) \\ v_\varepsilon\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) &\geqslant v_\varepsilon\left(\left(\left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \setminus \{f\}\right) \cup \{e\}\right) &&\Rightarrow v_\varepsilon(f) \geqslant v_\varepsilon(e) \end{aligned}$$

We investigate the cases $v_\delta(e) = v_\delta(f)$ and $v_\delta(e) > v_\delta(f)$ for the first inequality.

Case 1: $v_\delta(e) = v_\delta(f)$

- (i) As the weight function v is injective, we especially have $v(e) \neq v(f)$. The assumption can only be true if either ($f \in B_0$, $e \notin B_0$) or ($e \in B_0$, $f \notin B_0$) holds.
- (ii) With $\delta < \varepsilon$ we get the inequality $v_\varepsilon(e) \leqslant v_\varepsilon(f) \leqslant v_\delta(f) = v_\delta(e)$. In the case of $e \notin B_0$, we have $v_\delta(e) = v_\varepsilon(e)$ and – because of the inequality $v_\varepsilon(e) \leqslant v_\varepsilon(f) \leqslant v_\delta(f) = v_\delta(e)$ – also $v_\delta(f) = v_\varepsilon(f)$. But this means that $f \notin B_0$ and contradicts (i). So our assumption was wrong and $e \in B_0$ and $f \notin B_0$ holds.

Case 2: $v_\delta(e) > v_\delta(f)$
With $\delta < \varepsilon$ we get the inequality-chain $v_\varepsilon(e) \leqslant v_\varepsilon(f) \leqslant v_\delta(f) < v_\delta(e)$.

- (i) Because of $v_\varepsilon(e) < v_\delta(e)$ we know that $e \in B_0$.
- (ii) Assuming $f \in B_0$, we get $v_\delta(e) > v_\delta(f) \Leftrightarrow v_\varepsilon(e) > v_\varepsilon(f)$. This contradicts $v_\varepsilon(e) \leqslant v_\varepsilon(f)$ and shows that our assumption $f \in B_0$ is wrong. Thus, we have $e \in B_0, f \notin B_0$.

Thus, $e \in B_0$ holds for all $e \in \left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$.

Now, we choose an arbitrary $f \in \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right) \setminus \left(B_{1(\delta)} \cup B_{2(\delta)}\right)$. Then, Axiom $(V1')$ assures that some $e \in \left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$ exists which fulfills the equations (4.2.1) and (4.2.2).

Analogously to the investigations for an arbitrary $e \in \left(B_{1(\delta)} \cup B_{2(\delta)}\right) \setminus \left(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\right)$, the inequalities $v_\delta(e) \geqslant v_\delta(f)$ and

$v_\varepsilon(e) \leqslant v_\varepsilon(f)$ hold. We investigate the same two cases as above again. The only difference is that f is fixed instead of e. Again, we get $e \in B_0$ and $f \notin B_0$. That means that $f \notin B_0$ for all $f \in \big(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\big) \setminus \big(B_{1(\delta)} \cup B_{2(\delta)}\big)$.

Altogether, we proved that every element of the set $\big(B_{1(\delta)} \cup B_{2(\delta)}\big) \setminus \big(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\big)$ is part of B_0 and all elements of $\big(B_{1(\varepsilon)} \cup B_{2(\varepsilon)}\big) \setminus \big(B_{1(\delta)} \cup B_{2(\delta)}\big)$ are not part of B_0. This shows that every element of $\big(B_{1(\delta)} \cup B_{2(\delta)}\big) \setminus B_0$ also lies in $(B_{\varepsilon(1)} \cup B_{\varepsilon(2)})$. Thus, property (ii) holds. On the other hand, all elements of $B_0 \setminus \big(B_{1(\delta)} \cup B_{2(\delta)}\big)$ are not in $\big(B_{\varepsilon(1)} \cup B_{\varepsilon(2)}\big)$ and the property (i) holds. ■

Corollary 4.2.6.
Let M_v be a valuated matroid with an injective weight function $v : E \to \mathbb{R}^-$. Then, at most $rank(M)$ threshold parameters are possible.

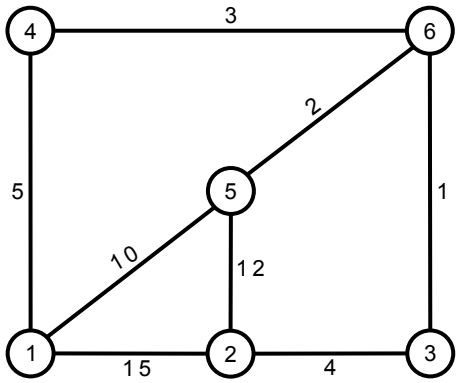
Proof
Theorem 4.2.5 gives us a structural monotonicity. That means that for every threshold parameter at least one element of the base B_0 falls out of the generated alternative and will not be part of any alternative which is optimal for a larger penalty parameter. On the other hand, every new element at a threshold parameter will be part of all optimal alternatives for any larger penalty parameter. The base B_0 contains $rank(M)$ elements. At every threshold parameter at least one of these elements falls out of the new alternative. Because of that, we can get at most $rank(M)$ threshold parameters. ■

Remark 4.2.7.
In the previous theorems and in Corollary 4.2.6 we used injective weight functions. All theorems and corollaries continue to hold for noninjective weight functions because we can modify this weight function to an injective one. This modification works analogously to Remark 3.2.4.

Example 4.2.8.
There are also examples with $rank(M)$ threshold parameters. Here the Example 3.2.5 from page 30 can be directly used. We can assume that the first alternative B_1 is always the optimal one. Then the second alternative is the alternative generated by the simple penalty method for the same penalty parameter. Such a pair of alternatives is optimal for any penalty parameter.

We showed that our pairs of alternatives fulfill the same properties for the mutual penalty method as for the simple penalty method. So the question is whether both methods generate the same alternatives for every penalty parameter.

Remark 4.2.9.
We show that the simple penalty method and the mutual penalty method can generate different alternatives. Therefore, we take a look at the following graph:



The following table shows which alternatives are optimal for the different intervals of penalty parameters (the edges are named by their weights):

$\varepsilon \in$	simple penalty alternatives	mutual penalty alternatives
$[0, 1]$	1,2,3,4,5 and 1,2,3,4,5	1,2,3,4,5 and 1,2,3,4,5
$[1, 2]$	1,2,3,4,5 and 1,2,3,4,10	1,2,3,4,5 and 1,2,3,4,10
$[2, 4]$	1,2,3,4,5 and 1,2,3,12,10	1,2,3,4,5 and 1,2,3,12,10
$[4, \infty]$	1,2,**3**,4,5 and 1,2,3,12,10	1,2,**15**,4,5 and 1,2,3,12,10

Table 4.2: Different Alternative Spanning Trees Generated by the Simple Penalty Method and the Mutual Penalty Method

The results for valuated matroids and shortest path problems show that punished elements can only be used if they are part of the optimal solution. The question is, whether this property holds for every optimization problem of sum type. In the following two sections we show that it does not hold for the assignment problem and the TSP.

4.3 Assignment Problem

4.3.1 A Fast Algorithm for the Generation of Mutual Assignments

In the previous two sections we gave fast algorithms to generate the pairs of mutual penalty alternatives for the shortest path problem and for valuated matroids. For the generation of mutual assignments we use the successive shortest path algorithm, as we already did for the shortest path problem. To use this algorithm, we build our graph as follows:

1. For every agent and every task we introduce a vertex, called $A_1, A_2, \ldots A_n$ and $T_1, T_2, \ldots T_n$.
2. Assigning agent A_i to task T_j incurs cost $w(i,j)$. So we insert an edge for all pairs (i,j) of agents and tasks with capacity one and weight $w(i,j)$.
3. It is possible that both generated alternatives assign an agent to the same task. Therefore we introduce our punished second edge for all pairs (i,j) of agents and tasks. These edges get capacity one and a weight of $(1+\varepsilon) \cdot w(i,j)$.
4. We also need a source s and a sink t. So we insert these two vertices s and t into our graph.
5. Every agent should be assigned to two tasks (one in every alternative) and every task should be performed by two agents. Thus, we introduce an edge of capacity two and weight zero from s to all agents and from all tasks to t.

The following figure shows the resulting graph. The values at the edges are their capacity and their weight (capacity, weight).

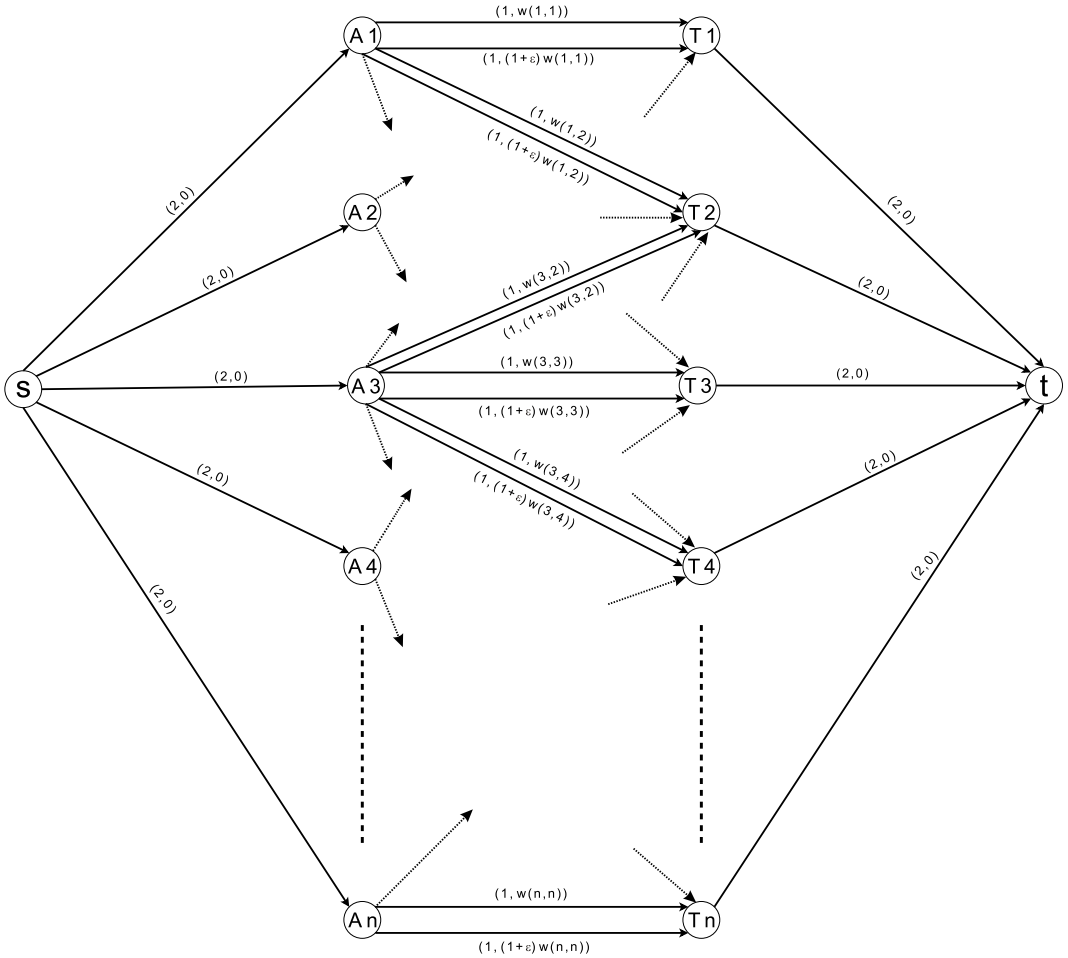


Figure 4.16: Transformation of a Mutual Assignment Problem to a Minimum Cost Flow Problem

The maximum flow in this flow network is $2n$. Using Dijkstra's algorithm to generate our shortest paths causes a runtime of $O(m \cdot log(n))$ per augmenting step. As we have to perform $2n$ augmenting steps, we get a total runtime of $O(mn \cdot log(n))$. The average runtime is faster, because many augmenting paths can be found very fast.

Every maximum flow gives us an assignment of the agents to perform two tasks (maybe twice the same task). On the other hand every task gets two agents which perform it. This is a necessary condition for two solutions of a assignment problem.

It remains to separate the solution of the maximum flow problem into two alternatives. Therefore, we use the following construction scheme:

1. Generate two empty sets $Assign_1$ and $Assign_2$. They will contain the assignments later. Mark all agents and tasks as not active and not fixed. Tasks and agents will be active if they are assigned to exactly one agent or task and they will be fixed if they are assigned to two agents or tasks.

2. Choose an arbitrary agent A_i which is not fixed so far. If no such agent exists, then STOP; $Assign_1$ and $Assign_2$ are our mutual penalty alternatives.

3. Let this agent A_i from step 2 be assigned to two tasks T_j and T_k. Insert the pair (A_i, T_j) to the set $Assign_1$ and the pair (A_i, T_k) to the set $Assign_2$. Mark the agent A_i as fixed. If $T_i \neq T_j$ mark both tasks as active and go to step 4, otherwise mark them as fixed and go to step 2.

4. Choose one of the active tasks T_l. So far T_l is part of one pair either in the set $Assign_1$ or in the set $Assign_2$. As this task is marked as active it is part of exactly one fixed pair (A_m, T_l). But it exists another agent A_o which performs it. If T_l is part of one pair in the set $Assign_1$ then insert the pair (A_o, T_l) to the set $Assign_2$, otherwise insert it to the set $Assign_1$. Mark the task T_l as fixed and not active, because now it is part of two assignments. If the agent A_o is marked as active, mark him as fixed and not active and go to step 2, otherwise mark it as active (and not fixed) and go to step 5.

5. Choose one of the active agents A_m. As A_m is marked as active it is part of exactly one fixed pair (A_m, T_l). But there exists another task T_o which is performed by A_m. If A_m is part of a pair in the set $Assign_1$ then insert the pair (A_m, T_o) to the set $Assign_2$, otherwise insert it to the set $Assign_1$. Mark the agent A_m as fixed and not active. If the task T_o is marked as active, mark it as fixed and not active and go to step 2, otherwise mark it as active and go to step 4.

After applying this scheme we have two assignments of the agents to the tasks. The scheme only uses the simple property that a task or an agent cannot be part of two pairs within the same assignment. Let A_i be assigned to T_j and T_k. If the pair (A_i, T_j) is part of one assignment, then the pair (A_i, T_k) must be part of the other assignment.

4.3.2 Properties of Mutual Assignments

For assignment problems it can be shown that a new alternative does not have to use new elements. In contrast to the shortest path problem and to the valuated matroids elements can be used by both alternatives although they are not part of the optimal solution. We show this with the help of the following example.

Example 4.3.1.
We give the costs of our assignment problem by the following table. The i-th row denotes agent i and the j-th column denotes task j. The "$-$" denotes that this assignment is not possible, it can also be understood as an infinite weight.

	1	2	3	4
1	10	$-$	$-$	1
2	50	5	$-$	$-$
3	200	$-$	10	$-$
4	$-$	100	100	10

In this problem there exist three possible assignments:

$$\begin{aligned} I &= \{(1,1),(2,2),(3,3),(4,4)\}, \\ II &= \{(1,4),(2,1),(3,3),(4,2)\} \text{ and} \\ III &= \{(1,4),(2,2),(3,1),(4,3)\}. \end{aligned}$$

The following table contains the weights of all possible pairs of assignments. As the pair (x, y) equals the pair (y, x) some cells are left blank:

	I	II	III
I	$70 + 35\varepsilon$	$196 + 10\varepsilon$	$341 + 5\varepsilon$
II		$322 + 161\varepsilon$	$467 + \varepsilon$
III			$612 + 306\varepsilon$

Thus, we have:

(I and I) is optimal for $0 \leqslant \varepsilon \leqslant 5.04$,
(I and II) is optimal for $5.04 \leqslant \varepsilon \leqslant 29$,
(I and III) is optimal for $29 \leqslant \varepsilon \leqslant 31.5$ and
(II and III) is optimal for $\varepsilon \geqslant 31.5$.

All other pairs of solutions cannot be optimal for any $\varepsilon \geqslant 0$. They are dominated by the pair (I and I).

For $\varepsilon = 31.5$ we get the pair (II and III) as new alternative. This alternative does not use any new couple of agents and tasks because all elements of II were also part of the solution (I and II) which was new at $\varepsilon = 5.04$. Analogously, all elements of III were part of the solution (I and III) which was new at $\varepsilon = 29$. Thus, a new alternative for the assignment problem does not have to use a new cell in the cost matrix.

For $\varepsilon \geqslant 31.5$ the alternative (II and III) is optimal. This alternative uses the element $(1, 4)$ twice. Thus, the assignment problem can also use elements in both alternatives, although they were not part of the optimal solution.

4.4 Directed Traveling Salesperson Problem

Analogously to the assignment problem, we show that a new pair of alternatives for the TSP does not have to use a new edge. Again, edges can be used in both alternatives, although they are not part of the optimal solution pair.

Example 4.4.1.
In the following table the entry (i, j) denotes the distance from city i to city j. A "–" in the table denotes that this edge does not exist.

	1	2	3	4	5	6
1	–	10	–	–	–	1
2	100	–	5	–	–	–
3	–	100	–	10	100	–
4	50	–	–	–	10	–
5	–	–	100	200	–	10
6	10	150	–	100	–	–

In this ensemble exist four different Hamilton cycles:

$$\begin{aligned} I &= 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 1, \\ II &= 1 \to 2 \to 3 \to 5 \to 6 \to 4 \to 1, \\ III &= 1 \to 6 \to 2 \to 3 \to 5 \to 4 \to 1 \text{ and} \\ IV &= 1 \to 6 \to 4 \to 5 \to 3 \to 2 \to 1. \end{aligned}$$

The following table contains the weights of all possible pairs of alternatives. As the pair (x, y) equals the pair (y, x) some cells are left blank:

	I	II	III	IV
I	$110 + 55\varepsilon$	$330 + 25\varepsilon$	$561 + 5\varepsilon$	$466 + 10\varepsilon$
II		$550 + 275\varepsilon$	$781 + 155\varepsilon$	$686 + 100\varepsilon$
III			$1012 + 506\varepsilon$	$917 + \varepsilon$
IV				$822 + 411\varepsilon$

For the different intervals of penalty parameters, we get

(I and I) is optimal for $0 \leqslant \varepsilon \leqslant 7.\overline{3}$,
(I and II) is optimal for $7.\overline{3} \leqslant \varepsilon \leqslant 9.0\overline{6}$,
(I and IV) is optimal for $9.0\overline{6} \leqslant \varepsilon \leqslant 19$,
(I and III) is optimal for $19 \leqslant \varepsilon \leqslant 89$ and
(IV and III) is optimal for $\varepsilon \geqslant 89$.

All other pairs are not optimal for any $\varepsilon \geqslant 0$, because they are dominated by the pair of alternatives (I and I).

For $\varepsilon = 89$ we get (IV and III) as new pair of alternatives. This pair of alternatives does not use any new edge, because all elements of IV were also part of the solution (I and IV), which was new at $\varepsilon = 9.0\overline{6}$. Analogously, all elements of III were part of the solution (I and III) which was new at $\varepsilon = 19$. Thus, a new pair of alternatives for the TSP does not have to use one or more new edges.

For $\varepsilon \geqslant 89$ the pair of alternatives (IV and III) is optimal. This pair uses the edge $(1, 6)$ twice. This shows that the TSP can also use elements in both alternatives, although they were not part of the optimal solution.

Chapter 5

Conclusions and Discussion

We investigated two different methods for the generation of alternative solutions – the simple penalty method and the mutual penalty method.

Regarding the simple penalty method, we used a structural monotonicity for valuated matroids to show that the maximum number of threshold parameters is exactly the rank of our matroid. For the shortest path problem on arbitrary multigraphs we found an upper bound of $n^2 - 5n + 10$ and a lower bound of $\frac{n^2}{4} - 1$ for the maximum number of threshold parameters. For the assignment problem and the TSP we gave lower bounds of $\frac{n^2}{4} - 1$ and $\frac{(n-1)^2}{4} - 1$, respectively for the maximum number of threshold parameters. The symbol n denotes the number of vertices or the number of agents and tasks.

The main research interest of this thesis is the mutual penalty method. Here we formulated some structural properties for the shortest paths and the valuated matroids. For these two optimization problems we showed that all common elements of the pair of alternatives are also part of the optimal solution. This property does not hold for the assignment problem and the TSP. Moreover, we proved a structural monotonicity for valuated matroids. With the help of this structural monotonicity and an example for the MST problem we showed that the maximum number of threshold parameters is exactly $rank(M)$. For the shortest path problem we showed that both alternatives use their common vertices in the same order. Furthermore, vertices that lie on the shortest $s-t$-path are used in the same order in all pairs of alternatives. With the help of these properties we gave an upper bound of $\frac{n^2}{2} - \frac{n}{2}$ for the number of threshold parameters. By giving an example with $\frac{n^2}{4} - 1$ threshold parameters we showed that the maximum number of threshold parameters lies in the class $\Theta(n^2)$.

Finding fast algorithms for the generation of mutual penalty alternatives was another research interest. For all investigated problems except the TSP we gave fast algorithms for the generation of mutual penalty alternatives.

At the beginning of this thesis we recapitulated the k-best method and the generalized penalty method as two other approaches for the generation of alternatives. For these approaches the number of alternatives or threshold parameters is already known. For the k-best method we simply set all edge weights to different powers of two and get all solutions as k-best alternatives for different k's. In this way, it only remains to find the number of different solutions for our optimization problems:

- For the MST problem, Cayley's formula (see [AZ 2003, pp. 173-178]) gives the tight upper bound of $n^{(n-2)}$.
- For the shortest path problem we get all permutations of all subsets of interior vertices as possible paths (if we forbid circuits). The number of these paths is $\sum_{i=0}^{n-2} \binom{n-2}{i} \cdot i! = (n-2)! \cdot \sum_{i=0}^{n-2} \frac{1}{i!} \approx (n-2)! \cdot e$.
- For the assignment problem with n tasks and n agents the maximum number of alternatives is $n!$
- For the TSP the maximum number of alternatives is $(n-1)!$

Thus, all investigated optimization problems give us an exponential number of k-best alternatives.

The known results for the generalized penalty method come from different research areas like multicriteria optimization, parametric optimization or sensitivity analysis. In the book by Gonzalez [Gon 2007] many references to results concerning the maximum number of threshold parameters can be found at the pages 30.9 and 30.10. So Gusfield [Gus 1980] and Carstensen [Car 1983] showed that the maximum number of threshold parameters for the shortest path problem lies in the class $n^{\Theta(log(n))}$. Eppstein [Epp 1998b] gave an $O(m \cdot n^{\frac{1}{3}})$ upper bound and an $\Omega(m \cdot \alpha(n))$ lower bound for the number of threshold parameters for the MST problem. Here $\alpha(n)$ denotes the inverse Ackermann function. For matroids, Eppstein showed that the maximum number of threshold parameters lies in the class $\Theta(|E| \cdot (rank(M))^{\frac{1}{3}})$.

Concerning the maximum number of threshold parameters we get the following results:

	k-best	Generalized Penalty	Simple Penalty	Mutual Penalty
Valuated Matroids	$EXP(\|E\|)$	$\Theta(\|E\| \cdot (rank(M))^{\frac{1}{3}})$	$rank(M)$	$rank(M)$
MST Problem	$EXP(n)$	$O(m \cdot n^{\frac{1}{3}})$ and $\Omega(m \cdot \alpha(n))$	$n-1$	$n-1$
Shortest Path Problem	$EXP(n)$	$n^{\Theta(log(n))}$	$\Theta(n^2)$	$\Theta(n^2)$

Chapter 6

Open Problems

1. Which upper bounds can be shown for the assignment problem and the TSP? Does the maximum number of threshold parameters also lie in the class $\Theta(n^2)$ for the simple penalty method and for the mutual penalty method?

2. Sameith [Sam 2005] studied the advantage of having two alternatives. Therefore, he generated two alternatives for the model problem with the help of the different penalty methods. Then, he applied a perturbation model to simulate troubles like traffic jams. After that, he chose the alternative which was the best one regarding to the perturbed weights. How good is it to have all ε-alternatives instead of only one?

3. For large problem instances it might be helpful to have not $\Theta(n^2)$ alternative to choose amongst, but for example $\Theta(log(n))$. Are there good methods to reduce the number of alternatives automatically? Sameith [Sam 2005] showed that the penalty parameter $\varepsilon = 0.2$ is very good for our scenario. Thus, it might be a good approach to choose more alternatives for small penalty parameters.

4. In [Sam 2005] it was studied which penalty parameter should be chosen to get good alternatives for different perturbations. Do we get better results if we specify how different both alternatives should be from each other? An approach for the simple penalty method could be to minimize $w(B_\varepsilon)$ subject to
$$\frac{w(B_\varepsilon \cap B_0)}{w(B_0)} \leqslant x.$$
The parameter ε depends strictly on x. As $w(B_\varepsilon \cap B_0)$ decreases in ε and $w(B_0)$ is constant, the fraction decreases in ε. On the other hand the weight $w(B_\varepsilon)$ increases in ε. Thus, ε will be chosen as small as possible subject to our condition.
How should x be chosen to get the best results?

5. If we want to get more than two alternatives, then it might not be the best approach to generate the penalty alternatives for different penalty parameters. A better approach is to apply the simple penalty method iteratively (see also [Sch 2003, pp. 19-20]) or to use the k-mutual penalty method. Here the k-mutual penalty method has monotonicity properties (Theorem 2.2.8 at page 12). The simple penalty method does not have such properties.
   (a) How many threshold parameters are possible if we generate three, four, ... alternatives in this way?
   (b) How should the penalty parameter be chosen to get the best k-set of alternatives?
   (c) By using the mutual penalty method to generate more than two alternatives, it is not clear how elements should be punished, which are part of three, four, ... alternatives. What are good settings for these punishments (for example sum of pairwise penalties)?
   (d) Which k is suitable to get good alternatives within suitable time?
   (e) The k-mutual penalty method generates a set of elements and we have the freedom to partition this set into k solutions. There are different approaches which could be applied:
      (i) $lexmin(w(B_1), w(B_2), \ldots w(B_n))$
      (ii) $min\ w(B_k)$ with $w(B_1) \leqslant w(B_2) \leqslant w(B_3) \leqslant \ldots \leqslant w(B_k)$
      (iii) $max\ w(B_1)$ with $w(B_1) \leqslant w(B_2) \leqslant w(B_3) \leqslant \ldots \leqslant w(B_k)$
      (iv) $min\ w(B_k) - w(B_1)$ with $w(B_1) \leqslant w(B_2) \leqslant w(B_3) \leqslant \ldots \leqslant w(B_k)$
      $\vdots$

      Which is the best approach amongst them?

      Walter [Wal 2009] investigated similar questions concerning the scheduling of independent jobs on identical machines. The partition problem described above could also be understood as a scheduling problem on identical machines. But here the jobs are not independent, because the partitions have to be solutions of the underlying sum type problem.

6. We can also use the penalty methods for problems which do not have the sum type structure. So minimum cost flow problems and transportation problems can be investigated analogously. How should we handle elements which appear more often in the optimal solution (for example an edge with flow k could be punished with factor $1 + k \cdot \varepsilon$)?
   Here, the monotonicity and convexity properties hold analogously. This can be shown analogously to the proofs of the monotonicity and convexity theorems in Chapter 2. We simply separate the weights into a punished part $p(B)$ and an unpunished part $w(B)$ and perform the proof in the same way as we did for sum type problems. In Appendix B we illustrate this with the help of the monotonicity theorem for the simple penalty method.

Appendix A

Known Bounds for the Simple Penalty Method

In this Appendix we repeat some results of the diploma thesis [Doe 2008].

A.1 Shortest Path Problem

In Section 3.1 we mentioned that there exist examples for the shortest path problem with $\frac{n^2}{4} - 1$ threshold parameters for the simple penalty method. In the following example we construct such a graph for $n = 6$ and then we generalize it for an arbitrary n.

Example A.1.1.
For an easier calculation, the weights are chosen in a way that all threshold parameters and weights are natural numbers. Also, every generated alternative should contain exactly one edge which is not part of the optimal solutions. Finally, we want that only alternatives of the following structure come up:

$$1 \to 2 \to \ldots (i-1) \to i \longrightarrow j \to (j+1) \to \ldots \to (n-1) \to n.$$

The idea for the case $n = 6$ is that the shortest path uses all vertices from 1 up to 6. Then, we cut the graph in the middle, that means between the vertices 3 and 4. Every alternative uses one of the $3 \cdot 3 = 9$ edges to connect these two parts. Table A.1 shows the alternatives that will be optimal for the different intervals of penalty parameters.

$\varepsilon \in$	path
$[0,1]$	$1 \to 2 \to 3 \to 4 \to 5 \to 6$
$[1,2]$	$1 \to 2 \to 3 \longrightarrow 5 \to 6$
$[2,3]$	$1 \to 2 \to 3 \longrightarrow 6$
$[3,4]$	$1 \to 2 \longrightarrow 4 \to 5 \to 6$
$[4,5]$	$1 \to 2 \longrightarrow 5 \to 6$
$[5,6]$	$1 \to 2 \longrightarrow 6$
$[6,7]$	$1 \longrightarrow 4 \to 5 \to 6$
$[7,8]$	$1 \longrightarrow 5 \to 6$
$[8,\infty]$	$1 \longrightarrow 6$

TABLE A.1: All 9 Alternative Paths for $n = 6$

We set the weights of the edges $(3,4)$, $(4,5)$, $(5,6)$ to one. For sake of simplicity, we want that every new alternative uses one length unit less of the original path than the precedent alternative. Thus, for $\varepsilon = 3$ the weight of the edge $(2,3)$ can be calculated as follows:

$$\begin{aligned} v\left(B_{[2,3]} \cap B_0\right) &= v\left(B_{[3,4]} \cap B_0\right) + 1 \\ v(1,2) + v(2,3) &= v(1,2) + v(4,5) + v(5,6) + 1 \\ v(2,3) &= 1 + 1 + 1 = 3 \end{aligned}$$

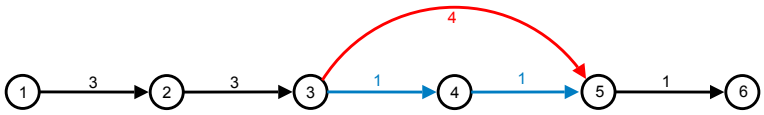
Analogously, it holds $v(1,2) = 3$.

Now, all weights of the solution B_0 are known. Thus, we have to calculate the weights of the remaining edges to get the alternatives as claimed in Table A.1.

At $\varepsilon = 1$ we have a threshold parameter. Here, the generated ε-optimal alternative switches from $1 \to 2 \to 3 \to 4 \to 5 \to 6$ to $1 \to 2 \to 3 \to 5 \to 6$. Thus, for $\varepsilon = 1$ we have:

$$\begin{aligned} (1+\varepsilon)\left[v(1,2) + v(2,3) + v(5,6)\right] + v(3,5) &= (1+\varepsilon)\left[v(1,2) + v(2,3) + v(3,4) + v(4,5) + v(5,6)\right] \\ v(3,5) &= (1+\varepsilon)\left[v(3,4) + v(4,5)\right] \\ v(3,5) &= 2\left[1+1\right] = 4 \end{aligned}$$

So far, we have the following graph which will be expanded below:



The remaining weights can be calculated analogously. We use the threshold parameters downwards in Table A.1. At these threshold parameters both involved

alternatives have the same punished weight. In this way, one equation has to be solved for every new edge. Step by step we get all weights for the graph G so that the alternatives claimed in Table A.1 are optimal for their according penalty parameters. The resulting graph of this construction is the following one:

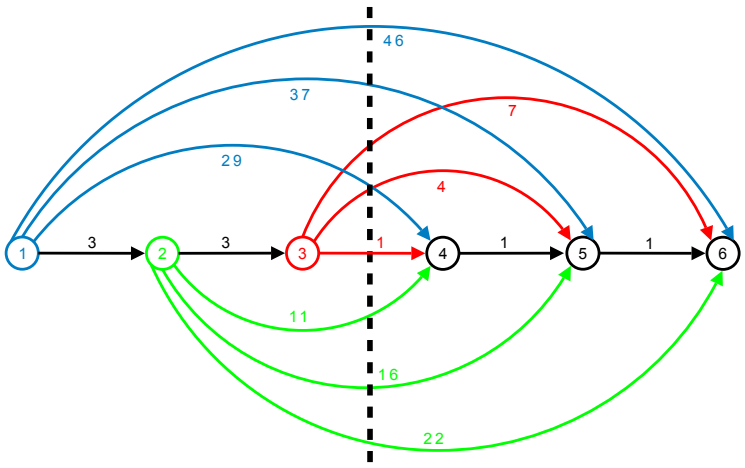


FIGURE A.1: An Example for the Shortest Path Problem with 8 Threshold Parameters

This example can be generalized to graphs with n vertices for an arbitrary n. Let the vertices be denoted with the numbers 1 up to n from the left to the right and let 1 be the starting vertex and n be the target vertex. We get $\frac{n^2}{4} - 1$ threshold parameters by assigning the following weights to the edges:

$$w(i,j) = \begin{cases} 1 & \text{if } j = i+1 \text{ and } i \geqslant \left\lfloor \frac{n}{2} \right\rfloor, \\ \left\lceil \frac{n}{2} \right\rceil & \text{if } j = i+1 \text{ and } i < \left\lfloor \frac{n}{2} \right\rfloor, \\ \frac{x_{i,j}(x_{i,j}+1)}{2} + 1 & \text{if } i \leqslant \left\lfloor \frac{n}{2} \right\rfloor,\ j > \left\lfloor \frac{n}{2} \right\rfloor \text{ and } j \geqslant i+2, \\ \infty & \text{otherwise,} \end{cases}$$

with $x_{i,j} := \left(\left\lfloor \frac{n}{2} \right\rfloor - i\right)\left\lceil \frac{n}{2} \right\rceil + j - \left\lfloor \frac{n}{2} \right\rfloor$.

The sequence of alternatives is analogous to the example for $n = 6$ as shown in Table A.1. All alternatives are identified by the edge passing the middle of the graph. This middle was symbolized by a dotted line so far.

In this sequence every alternative uses another (i,j) with $i \leqslant \left\lfloor \frac{n}{2} \right\rfloor$ and $j > \left\lfloor \frac{n}{2} \right\rfloor$. Furthermore all of these edges (i,j) are used by one alternative. This means that there exist $\left\lceil \frac{n}{2} \right\rceil \left\lfloor \frac{n}{2} \right\rfloor$ alternatives in this sequence. The number of threshold parameters is the number of alternatives minus one. Thus, we get $\left\lceil \frac{n}{2} \right\rceil \left\lfloor \frac{n}{2} \right\rfloor - 1 = \frac{n^2}{4} - 1$ as the number of threshold parameters in these examples.

The correctness of this example is ensured because all possible paths of the graph are simple penalty alternatives. Theorem 2.2.4 assures that the order of these alternatives cannot be switched, thus this sequence is correct.

A.2 Valuated Matroids

In this section we give a proof for Theorem 3.2.1.

Theorem 3.2.1.
Let E be a finite set. Assume $0 \leqslant m \leqslant |E|$ and let $v : E \to \mathbb{R}^-$ be an injective weight function on E and let the pair $M_v = (E, v)$ be a valuated matroid of rank m. Then, the following two properties hold:

(i) For all $\delta < \varepsilon$ and all $a \in B_0 \setminus B_\delta$ it holds $a \notin B_\varepsilon$.

(ii) For all $\delta < \varepsilon$ and all $b \in B_\delta \setminus B_0$ it holds $b \in B_\varepsilon$.

Proof
If B_ε and B_δ are equal then the properties (i) and (ii) hold. Thus, we assume that $B_\varepsilon \neq B_\delta$ holds. Now, we choose an arbitrary element $e \in B_\delta \setminus B_\varepsilon$. The modified Axiom $(V1')$ for valuated matroids of sum type assures that there exists an element $f \in B_\varepsilon \setminus B_\delta$ with

$$v_\delta(B_\delta) + v_\delta(B_\varepsilon) = v_\delta((B_\delta \setminus \{e\}) \cup \{f\}) + v_\delta((B_\varepsilon \setminus \{f\}) \cup \{e\}) \quad \text{(A.2.1)}$$

and

$$v_\varepsilon(B_\delta) + v_\varepsilon(B_\varepsilon) = v_\varepsilon((B_\delta \setminus \{e\}) \cup \{f\}) + v_\varepsilon((B_\varepsilon \setminus \{f\}) \cup \{e\}) \quad \text{(A.2.2)}$$

As defined, the alternative B_δ is optimal for the penalty parameter δ and B_ε is optimal for ε. Thus, we especially have:

$$\begin{aligned} v_\delta(B_\delta) &\geqslant v_\delta((B_\delta \setminus \{e\}) \cup \{f\}) &&\Rightarrow v_\delta(e) \geqslant v_\delta(f) \\ v_\varepsilon(B_\varepsilon) &\geqslant v_\varepsilon((B_\varepsilon \setminus \{f\}) \cup \{e\}) &&\Rightarrow v_\varepsilon(f) \geqslant v_\varepsilon(e) \end{aligned}$$

Concerning the first inequality we investigate the following two cases:

Case 1: $v_\delta(e) = v_\delta(f)$

(i) As the weight function is injective, we especially have $v(e) \neq v(f)$. The assumption can only be true if either ($f \in B_0$ and $e \notin B_0$) or ($e \in B_0$ and $f \notin B_0$).

(ii) With $\delta < \varepsilon$ we get $v_\varepsilon(e) \leqslant v_\varepsilon(f) \leqslant v_\delta(f) = v_\delta(e)$.
Assuming $e \notin B_0$ we get $v_\delta(e) = v_\varepsilon(e)$ and $v_\delta(f) = v_\varepsilon(f)$. This means that $f \notin B_0$ and contradicts (i). So our assumption was wrong and $e \in B_0$, $f \notin B_0$ holds.

Case 2: $v_\delta(e) > v_\delta(f)$
With $\delta < \varepsilon$ we get $v_\varepsilon(e) \leqslant v_\varepsilon(f) \leqslant v_\delta(f) < v_\delta(e)$.

(i) This inequality gives $v_\varepsilon(e) < v_\delta(e)$ and therewith $e \in B_0$.

(*ii*) Assuming $f \in B_0$, we get $v_\delta(e) > v_\delta(f) \Leftrightarrow v_\varepsilon(e) > v_\varepsilon(f)$. This contradicts $v_\varepsilon(e) \leqslant v_\varepsilon(f)$ and shows that our assumption $f \in B_0$ is wrong. Thus, we have $e \in B_0, f \notin B_0$.

Thus, $e \in B_0$ holds for all $e \in B_\delta \setminus B_\varepsilon$.

Now, we choose an arbitrary $f \in B_\varepsilon \setminus B_\delta$. Then Axiom $(V1')$ assures that some $e \in B_\delta \setminus B_\varepsilon$ exists with $(A.2.1)$ and $(A.2.2)$.

As for an arbitrary $e \in B_\delta \setminus B_\varepsilon$ the inequalities $v_\delta(e) \geqslant v_\delta(f)$ and $v_\varepsilon(e) \leqslant v_\varepsilon(f)$ hold. We investigate the same two cases as above again. The only difference is that f is fixed instead of e. The results of both cases are the same as for a fixed e and we get $e \in B_0$ and $f \notin B_0$.

This means that $f \notin B_0$ holds for all $f \in B_\varepsilon \setminus B_\delta$.

Altogether, we showed that every element of the set $B_\delta \setminus B_\varepsilon$ is part of the base B_0 and all elements of $B_\varepsilon \setminus B_\delta$ are not in B_0. On the one hand each element of $B_\delta \setminus B_0$ is also part of B_ε and statement (ii) holds. On the other hand all elements of $B_0 \setminus B_\delta$ are not part of B_ε and so the statement (i) holds. ■

Appendix B

Monotonicity Theorem for Other Optimization Problems

In this thesis we investigated the penalty methods for sum type problems. But it is possible to use these approaches also for other kinds of problems In this context, there exist some results regarding the generation of simple penalty alternatives for the scheduling of independent jobs on identical machines (see [Kae 2009]). Here, all jobs on the machine which causes the makespan could be punished.

Also for many other problem classes like network flow problems we can generate alternatives with the simple penalty method. Regarding the minimum cost flow problem (see Section 2.4) it is not clear, how elements should be punished. The problem is that the edges might have different flows and it makes sense to use a punishment which depends on the flows in the optimal solution. One possible approach could be to punish an edge e by multiplying its weight by $(1+f(e)\cdot\varepsilon)$. Here $f(e)$ denotes the flow along e in the optimal solution. Analogously, we can generate mutual penalty alternatives. For the minimum cost flow problem monotonicity and convexity properties exist analogously to the properties for sum type problems (Theorems 2.2.3, 2.2.4, 2.2.8 and 2.2.9). As an illustration we prove a monotonicity theorem for the simple penalty method. As this proof and all other proofs of the announced properties work analogously to the proofs in Chapter 2, we do not prove the convexity theorem and we also do not prove anything for the mutual penalty method.

Theorem B.1.1.
Let $w, p : E \to \mathbb{R}^+$ be positive weight functions on E. Let B_ε be a penalty alternative for $\varepsilon > 0$. Like for the generalized penalty method, $w(B)$ denotes the weight of the alternative B and $p(B)$ its punishment. This means $w_\varepsilon(B) = w(B) + \varepsilon \cdot p(B)$. Then the following statements hold:

(i) $p(B_\varepsilon)$ *is weakly monotonically decreasing in* ε.

(ii) $w(B_\varepsilon)$ *is weakly monotonically increasing in* ε.

Proof
Let δ and ε be two arbitrary nonnegative real numbers with $0 \leqslant \delta < \varepsilon$. As B_δ and B_ε are optimal alternatives for the penalty parameters ε and δ, the following inequalities hold:

(i) In the case $\varepsilon < \infty$ we have

$$w(B_\varepsilon) + \varepsilon \cdot p(B_\varepsilon) \leqslant w(B_\delta) + \varepsilon \cdot p(B_\delta), \tag{B.1.1}$$
$$w(B_\delta) + \delta \cdot p(B_\delta) \leqslant w(B_\varepsilon) + \delta \cdot p(B_\varepsilon). \tag{B.1.2}$$

Subtracting (B.1.2) from (B.1.1) we get

$$\begin{aligned} & (\varepsilon - \delta) \cdot p(B_\varepsilon) \leqslant (\varepsilon - \delta) \cdot p(B_\delta) \\ \Leftrightarrow \quad & p(B_\varepsilon) \leqslant p(B_\delta). \end{aligned} \tag{B.1.3}$$

In the case $\varepsilon = \infty$, inequality (B.1.3) follows directly from the definition of B_∞.

(ii) By subtracting (B.1.3) multiplied by δ from (B.1.2) we get $w(B_\varepsilon) \geqslant w(B_\delta)$.

■

Nomenclature

E	Base Set (often Edge Set)
V	Set of Vertices
$G = (V, E)$	Graph with Edge Set E and Vertex Set V
$\tilde{G}$	Reduced graph in the proofs of the upper bounds for the shortest path problem
S	Set of Feasible Solutions
$\mathcal{P}(E)$	Power Set of E
$\mathbb{R}$	Set of Real Numbers
$\mathbb{R}^+$	Set of Positive Real Numbers
$\mathbb{R}^-$	Set of Negative Real Numbers
w	Weight Function
f	Flow along an Edge
c	Capacity of an Edge
d	Shortest Path Distance [$d(i) =$ Distance from s to i] also Distance Function for the TSP
$O(f(n))$	Asymptotic Upper Bound of $f(n)$
$\Omega(f(n))$	Asymptotic Lower Bound of $f(n)$
$\Theta(f(n))$	Asymptotic Upper and Lower Bound of $f(n)$
$EXP(n)$	Exponential Function in n

$A = \{A_1, A_2, \ldots, A_n\}$	Set of Agents for the Assignment Problem
$T = \{T_1, T_2, \ldots, T_n\}$	Set of Tasks for the Assignment Problem
$C = \{C_1, C_2, \ldots, C_n\}$	Set of Cities for the TSP
min	This Function returns the smallest Value of the given Arguments
lexmin	Lexicographic Minimization. The returned Value is Minimal concerning the first Argument and amongst all these Solutions it is also minimal concerning the second Argument
s	Start Vertex
t	Target Vertex
n	Problem Size (Number of Vertices, Cities, Agents and Tasks)
m	Number of Edges
MST	Minimum Spanning Tree
TSP	Traveling Salesperson Problem
$x \to y$	Edge from x to y
$X \to Y$	Path from X to Y as used by the Shortest $s-t$-path

Penalty Methods

$\varepsilon,\ \delta$	Penalty Parameters
$\varepsilon', \varepsilon^*$	Threshold Parameters
p	Penalty Function for the Generalized Penalty Method
B	Solution of an Optimization Problem, also called Base for Valuated Matroids
B_0	Optimal Solution of an Optimization Problem
$B_\varepsilon, B_\delta, \ldots$	Penalty Alternatives for the Penalty Parameter $\varepsilon, \delta, \ldots$
B_∞	Optimal Solution for very large Penalty Parameters

$\{B_{1(\varepsilon)}, B_{2(\varepsilon)}\}$	Pair of Mutual Penalty Alternatives for the Penalty Parameter ε
$\{B_{1(\infty)}, B_{2(\infty)}\}$	Pair of Mutual Penalty Alternatives for very large Penalty Parameters
OPT_B	Set of Penalty Parameters for which the Solution B is an Optimal Penalty Alternative
$B_{[\varepsilon_i,\varepsilon_j]}$	Optimal Solution for $\varepsilon \in [\varepsilon_i, \varepsilon_j]$
$P_\varepsilon, P_0, P_\infty \ldots$	Penalty Alternatives for the Shortest Path Problem (see $B_\varepsilon, B_0, B_\infty, \ldots$)
$\mathcal{P}_{old}$	Set of Solutions that were Optimal for a Smaller Penalty Parameter
P_{old}	Solution that was Optimal for a Smaller Penalty Parameter
P_{new}	Solution that is New for the temporary investigated Threshold Parameter

Valuated Matroids

M_v	Valuated Matroid
v	Valuation of Valuated Matroids
$\binom{E}{m}$	Set of all m-subsets of E
$rank(M_v)$	Rank of the Valuated Matroid M_v
$\mathfrak{B}_v$	Base Set of the Valuated Matroid M_v

Bibliography

[ABS 2002] I. Althöfer, F. Berger and S. Schwarz: Generating True Alternatives with a Penalty Method, technical report, http://www.minet.uni-jena.de/Math-Net/reports/sources/2002/02-04report.pdf (2002).

[Alt 1998] I. Althöfer: 13 Jahre 3-Hirn – Meine Schach-Experimente mit Mensch-Maschinen-Kombinationen, 3-Hirn-Verlag, http://www.3-hirn-verlag.de (1998).

[Alt 1999] I. Althöfer: Decision Support Systems with Multiple Choice Structure, technical report, http://www.minet.uni-jena.de/Math-Net/reports/sources/1999/99-31report.ps (1999).

[Alt 2000] I. Althöfer: Multiple Choice Systems for Decision Support, technical report, http://e-pub.uni-weimar.de/volltexte/2005/593/pdf/094p_Althoefer.pdf (2000).

[AMO 1993] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River, NJ (1993).

[AND 1997] AND Software GmbH. Car routing program "Route Germany" (1997).

[AW 1999a] I. Althöfer and W. Wenzel: Two-Best Solutions under Distance Constraints: The Model and Exemplary Results for Matroids, Advances in Applied Mathematics 22, pp. 155-185 (1999).

[AW 1999b] I. Althöfer and W. Wenzel: k-Best Solutions under Distance Constraints in Valuated Δ-Matroids, Advances in Applied Mathematics 22, pp. 381-412 (1999).

[AZ 2003] M. Aigner, G. M. Ziegler: Proofs from the Book, Springer-Verlag, Berlin, 3rd Revised Edition (2003).

[Bel 1958] R. Bellman: On a routing problem. Quarterly of Applied Mathematics, 16, pp. 87-90 (1958).

[Ber 2000] F. Berger: K alternative instead of k shortest paths, diploma thesis, Faculty of Mathematics and Informatics, Friedrich-Schiller-University Jena (2000).

[BK 1958] R. Bellman, R. Kalaba: On kth best policies, The RAND Corporation, Paper P-1417 (1958).

[Car 1983] P. Carstensen: The Complexity of Some Problems in Parametric Linear and Combinatorial Programming, PhD thesis, Univ. of Michigan (1983).

[Cay 1897] A. Cayley, The Collected Mathematical Papers of Arthur Cayley, Vol. XIII, Cambridge University Press, Cambridge, Uk (1897).

[CLR 1990] T. H. Cormen, C. E. Leiserson, R. L. Rivest: Introduction to Algorithms, The MIT Press (1990).

[CW 2004] K. Chao, B. Y. Wu: Spanning Trees and Optimization Problems, Chapman & Hall / CRC (2004).

[Din 1970] E. A. Dinic: Algorithm for solution of a problem of maximum flow in a network with power estimation, Soviet Math. Doklady, Vol 11, pp. 1277-1280 (1970).

[Doe 2008] M. Dörnfelder: Maximale Anzahlen von Sprungstellen bei der Alternativen-Generierung für Kürzeste-Wege-Probleme und bewertete Matroide (in German), diploma thesis, Faculty of Mathematics and Informatics, Friedrich-Schiller-University Jena (2008).

[DW 1990] A. W. M. Dress and W. Wenzel: Valuated matroids - A new look at the greedy algorithm, Applied Mathematics Letters, Volume 3, Issue 2, pp. 33-35 (1990).

[Edm 1965] J. Edmonds: Minimum Partition of a Matroid into Independent Subsets, Journal of Research of the National Bureau of Standards, 69B, pp. 67-72 (1965).

[Ehr 2005] M. Ehrgott: Multicriteria Optimization, Second Edition, Springer (2005).

[EK 1972] J. Edmonds and R. M. Karp: Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM 19 (2), pp. 248-264 (1972).

[EP 1992] V. Emelichev and V. Perepelitsa: On cardinality of the set of alternatives in discrete many-criterion problems, Discrete Mathematics and Applications, 2(5), pp. 461-471 (1992).

[Epp 1998a] D. Eppstein: Finding the k shortest paths. SIAM Journal of Computing, Vol. 28, No. 2, pp. 652-673 (1998).

[Epp 1998b] D. Eppstein: Geometric lower bounds for parametric matroid optimization, Discrete and Computational Geometry, Volume 20, Number 4 (1998).

[FF 1956] L. R. Ford, D. R. Fulkerson: Maximal flow through a network, Canadian Journal of Mathematics 8: 399-404 (1956).

[Gon 2007] T. F. Gonzalez: Handbook of Approximation Algorithms and Metaheuristics, Chapman and Hall/CRC Computer and Information Science Series (2007).

[Gus 1980] D. Gusfield: Sensitivity Analysis for Combinatorial Optimization, Ph.D. thesis, University of California, Berkeley (1980).

[Han 1979] P. Hansen: Bicriterion path problems, In G. Fandel and T. Gal, editors, Multiple Criteria Decision Making Theory and Application, volume 177 of Lecture Notes in Economics and Mathematical Systems, pp. 109-127, Springer Verlag, Berlin (1979).

[HK 2000] H.W. Hamacher, K. Klamroth: Lineare und Netzwerkoptimierung - Linear and Network Optimization, Bilingual Text Book, Vieweg (2000).

[HR 1994] H. W. Hamacher and G. Ruhe: On spanning tree problems with multiple objectives, Annals of Operations Research, 52, pp. 209-230 (1994).

[Kae 2009] N. Kästner: Alternativen-Generierung für das Load-Balancing Problem (in German), term paper, Friedrich-Schiller-University Jena, in preparation (2009).

[Knu 1968] D. E. Knuth: The Art of Computer Programming, Addison-Wesley, Reading, MA, Vol. 1 (1968).

[KT 2005] J. M. Kleinberg, E. Tardos: Algorithm Design, Addison Wesley (2005).

[Kue 1998] K. Küfer: On the asymptotic number of efficient vertices in multiple objective linear programming, Journal of Complexity, 14, 333-377 (1998).

[Meh 1984] K. Mehlhorn: Data structures and algorithms 1,2 and 3, Springer (1984).

[MMW 2007] M. Maier, S. Mecke and D. Wagner: Algorithmic Aspects of Minimum Energy Edge-Disjoint Paths in Wireless Networks, Lecture Notes In Computer Science Vol. 4362, Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science (2007).

[Nas 1964] C. St. J. A. Nash-Williams: Edge-disjoint spanning trees of finite graphs, Journal of the London Mathematical Society 36, pp. 445-450 (1964).

[Nyb 2002] M. A. Nyblom: Some Curious Sequences Involving Floor and Ceiling Functions, The American Mathematical Monthly, Vol. 109, No. 6, pp. 559-564 (2002).

[Rec 1989] A. Recski: Matroid theory and its applications in electric network theory and in statics (Algorithms and Combinatorics, Vol. 6) Springer Verlag, Berlin New York and Akadmiai Kiad, Budapest (1989).

[RT 1985] J. Roskind and R. E. Tarjan: A note on finding minimum-cost edge-disjoint spanning trees, Mathematics of Operations Research 10, pp. 701-708 (1985).

[Sam 2005] J. Sameith: Generating Alternative Solutions for Discrete Optimization Problems with Uncertain Data. 3-Hirn-Verlag, http://www.3-hirn-verlag.de. Also doctoral dissertation, Faculty of Mathematics and Informatics, Friedrich-Schiller-University Jena (2005).

[Sch 2003] S. Schwarz: On the Generation of Multiple Candidate Solutions for Discrete Optimization Problems, doctoral dissertation, Faculty of Mathematics and Informatics, Friedrich-Schiller-University Jena (2003).

[ST 1984] J. W. Suurballe and R.E. Tarjan: A quick method for finding shortest pairs of disjoint paths, Networks 14 (1984).

[Suu 1974] J. W. Suurballe: Disjoint paths in a network, Networks, Vol.4, pp. 125-145 (1974).

[Tar 1983] R. E. Tarjan: Data Structures and Network Algorithms, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA, (1983).

[Tut 1961] W. T. Tutte: On the problem of decomposing a graph into n connected factors, Journal of the London Mathematical Society 36, pp. 221-230 (1961).

[Wal 2009] R. Walter: A Manifold Analysis of Scheduling Independent Jobs on Identical Machines, doctoral dissertation, Faculty of Mathematics and Informatics, Friedrich-Schiller-University Jena, in preparation (2009).

[Wel 1976] D. J. A. Welsh: Matroid Theory, Academic Press (1976).

Ehrenwörtliche Erklärung

Hiermit erkläre ich:

- dass mir die Promotionsordnung der Fakultät für Mathematik und Informatik der Friedrich-Schiller-Universität Jena bekannt ist,
- dass ich die Dissertation selbst angefertigt habe und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben sind,
- dass die Hilfe eines Promotionsberaters nicht in Anspruch genommen wurde und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe und
- dass ich die gleiche, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung nicht bei einer anderen Hochschule als Dissertation eingereicht habe.

Jena, 18.12.2009

Tabellarischer Lebenslauf

Persönliche Daten

Name:	Martin Dörnfelder
Geburtsdaten:	1.10.1984 in Sondershausen
Anschrift:	Mühlenstraße 51 07745 Jena
Telefon:	0176 / 40166407
E-Mail:	martin_doernfelder@gmx.de
Familienstand:	ledig

Schulbildung

09/1995 – 06/2003	Geschwister-Scholl-Gymnasium in Sondershausen Abschluss: Abitur

Studium

10/2004 – 09/2008	Studium der Mathematik an der Friedrich-Schiller-Universität Jena Abschluss: Diplom
10/2008 – 12/2009	Promotion an der Friedrich-Schiller-Universität Jena

Arbeitsverhältnisse

10/2008 – 12/2009	Wissenschaftlicher Mitarbeiter am Institut für Angewandte Mathematik der Friedrich-Schiller-Universität Jena